ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO INDUSTRIAL

Título del proyecto:

OPTIMIZATION OF A PRODUCTION AND ASSEMBLY PROCESS

Aitana Azcona Iriarte

Fermín Fco. Mallor Giménez

Pamplona, Octubre 2015

Acknowledgement

I am highly indebted to Fermín Mallor for his guidance and constant support, always encouraging me to not give up on the project and helping me in everything he could.

Very much appreciation towards the Precise workers with whom I have worked in the quality department, especially Jörn Sieberg, which has always given me his help in what I asked for.

I would like to thank my parents, Ángel and Ana, who at any time give me what I need: support, understanding, encouragement and strength; Jesús, for his help, his patience and support especially in this difficult year; my three friends, María B., María E. and Claudia, who are more than friends to me, essential people in my life and that despite being far from me, they have always been there.

Finally, I would like to extend thanks towards my family and college friends.

Abstract

The aim of this project is the resolution of a specific problem that usually arises in some manufacturing companies within their production processes. The lack of an appropriate procedure to follow with regard to allocating and matching parts occasionally causes an inefficient forethought in the purchase of such parts, affecting the production process and its efficiency. Most of the times this also means a greater economic cost than the strictly necessary one. What is sought, therefore, is the development of a procedure to solve such problems.

The emphasis is to be placed on the following aspects of the project:

- development of a mathematical procedure to solve the problems explained above in a theoretical way
- implementation of that process for the resolution of the problem in a practical way
- results obtained by the implementation

The results obtained show that the existing problem can be solved in a accurate and efficient manner and that the company can benefit from the implementation of the procedure developed.

Contents

List of Figures

Chapter 1

Introduction

1.1 Description of the company

1.1.1 Fischer's Group

Fischer group's history begins in 1939, when Stohler & Fischer AG was founded in Inkwil, Switzerland. The company was focused in the production of precision spindles, being the manufacture of grinding spindles a key procedure within its whole production process

In 1952 the distribution of the company Stohler & Fischer into two autonomous companies by Ernst Fischer takes place: W. Stohler is located in Inkwil (Switzerland) and E. Fischer AG, the factory of grinding spindle, in Herzogenbuchsee (Switzerland).

After some relocations in 1953 the production of grinding spindles and its sales department was moved to a new building, which will later become the current headquarters of the firm, in Oberönz, Switzerland.

In 1997 and 2001, two subsidiaries are based. The first one is Fischer USA Spindle Technologies, Inc. founded in New Brighton, Minnesota, USA and the second one is Fischer Europe Service S.A.R.L. founded in Peillonnex, France.

In 2002 the process-oriented management information system is certified according to ISO 9001:2000.

Between 2003 and 2006 four takeovers occur:

- Takeover of Fortuna-Werke and founding of the subsidiary FISCHER FORTUNA GmbH in Weil der Stadt, Germany.
- Takeover of Evotech AG and change of the company's name to Fischer Engineering Solutions AG, in Switzerland.
- Takeover of The Precise Corporation, USA
- Takeover of Precise Präzisionsspindeln GmbH, Germany

In 2009, the holding company FISCHER Spindle Group AG is founded at the headquarters in Herzogenbuchsee (Switzerland). Shortly after PRECISE Technologies GmbH is founded.

Since 2014 the Company operates under the new name FISCHER Spindle Group AG. Its brand and image is modernized for its 75th anniversary.

Nowadays, this group has four manufacturing locations for precision spindles around the world where they currently employ a total of approximately 300 people. They are on the market for nearly 80 years now, where they have an international presence with sales and service.

The main industries for which the Fischer's group produce are very diverse. Mainly they are:

- Automotive Industry

  The automobile industry usually uses milling spindles throughout its interlinked machining centres. Typical applications include the machining of gearbox housings, engine blocks, and cylinder heads, among others. A continuous stop-start operation with multiple tool changes each minute demands that the spindle accelerates and brakes quickly for short tool-change cycle times. The great number of differing drilling and milling operations require optimal axial and radial rigidity of the spindle, while 365-day, around-the-clock operation demands spindle systems of the very highest reliability.

- Watch Industry

  The watch industry demands the same specifications explained above

with regard machining spindles. Machining depends on the smallest tools, high rotation speeds, and the utmost precision. To create the finest threads, an exact position resolution of the spindle rotor bearing is required -only this way can the feed axis of the machine work together with the spindle, like clockwork.

- Aerospace

  The aerospace industry primarily mills filigree fin components from aluminum with a material removal rate of up to 98 percent. The cost-effectiveness of the milling process is determined by the material removal rate (MRR) of the milling machine. The milling spindle is the most heavily used machine element and it deserves the greatest attention. In addition, the material removal rate is essential to the high profitability of this industrial sector. Likewise, machine availability - 24 hours a day, preferably 365 days a year - is paramount: dependability matters.

- Railway

  The cost-effectiveness of the milling process is determined by the material removal rate and the process accuracy of the production plant. This requires compact milling spindles of the utmost quality, precision, and performance parameters.

- Electronics

  Smart phones, tablets or PCs are subject to short production cycles. No sooner than one is the proud owner of a new gadget, and its successor is already in the shops. Style and design are critical factors. Shiny surfaces make the smallest dimensional deviation visible. The accuracy and finish achievable during production are crucial!

1.1.2 Precise

"Innovative technology leader in spindle production for machine tools with a focus on high rotation speed, maximum precision and strong performance with a compact size of up to Φ *100 mm"*

This is how the Fischer's group described PRECISE, one of his subsidiaries in Germany.

Since 1933, PRECISE Technologies GmbH has been developing and distributing products under the PRECISE brand name. It offers a range of spindle solutions for tool and mould fabrication applications, the optical industry, the fields of medical and dental technology, as well as for the micro chipping and aviation sectors. Its core competences are components and fabricated elements with external diameters of up to 120 millimetres. PRECISE Technologies specialises in spindles with integrated high-frequency motors using synchronous or asynchronous technology, and delivering speeds of up to 160,000 rotations per minute (rpm). PRECISE spindles are designed for high-precision fabrication with tolerances of less than a micrometre, and represent the ideal solution for precision milling, drilling, and grinding applications in which small spindles are indispensible.

In its factory located in Langenfeld (Germany), which employs about 40 people, there is an annual turnover of around 6-7 million euro.

In regards to the market the options are vast, since as it has already been said the products produced are used in:

- Precision Machining
- Optical Industry
- Dental industry
- Automotive Industry
- Watch Industry
- Medical technology
- Tool Mould

The existence of so many potential markets requires a constant development carried out by the company, since a similar evolution of its competitors to remain competitive is needed.

Precise Technologies operates in three different segments:

- Manufacture of new spindles

  The FISCHER Spindle Group offers contract manufacturing of high-precision, rotationally symmetrical components and assemblies in small and medium quantities.

- Spindles repair

  The worldwide service network ensures short lead times for spindle repairs. The benefit from the large inventory of original replacement parts as well as of the continuous improvement process over the lifetime of the product. They execute the same level of accuracy just as if the spindle were being built new.

- Spindle commissioning

  On site spindle commissioning by a FISCHER Spindle service engineer. Assistance with the installation of the spindle, inspection and calibration of peripheral settings, commissioning and test acceptance. All work is executed with a detailed checklist for quality control and data tracking.

1.2 Project summary

This project analyses the main problems that take place in a company, Precise Technologies, specifically within its production process and proposes a methodology to solve them. Furthermore, this methodology is implemented in a computer's software, which is tested by using real data. The results shows its good performance.

The final product produced by the company is a spindle, which is composed of several parts. Among these parts that compose the spindle, some are bought and some others produced right at the factory.

The problems arise when it comes to producing the parts and to choosing them to create the spindle. The firm currently lacks a standardised procedure to allocate such parts, so such allocation must be done personally by a worker. This brings along some problems, such as:

- A waste in the number of purchased parts.
  Many of the pieces bought are finally not used, representing a significant cost for the company. This occurs because the manufactured pieces, the pieces that at the end of the process are assigned with the purchased parts to obtain the spindle, are not manufactured according to these purchased parts.

- The company lacks a method to assign the pieces together, meaning that the assignment is completely random.
  This means it's impossible to predict the number of pieces of each element needed to make a certain number of spindles, the time it will take ...

Solving these problems can result in significant improvements for the company, and this is why it is considered an interesting project with a view to its implementation.

The maximum utilization of purchased parts would be achieved when the parts that are manufactured are produced depending on those purchased. However this is not the case so far.

The procedure to be developed should allow the company the calculation of the optimum nominal dimension of the elements that later are fitted with the purchased ones.

Furthermore, the procedure is also aimed at stablish the assignments between the parts produced and purchased, so that the largest number of optimal spindles is obtained at the end of the production process.

It is important to emphasize that the aim of the project is not just the solution of the problem but also the obtainment of the best achievable results.

Figure 1 shows a very simplified figure of the spindle, more specifically the elements on which the project is focused, as the final spindle is composed of many more elements.

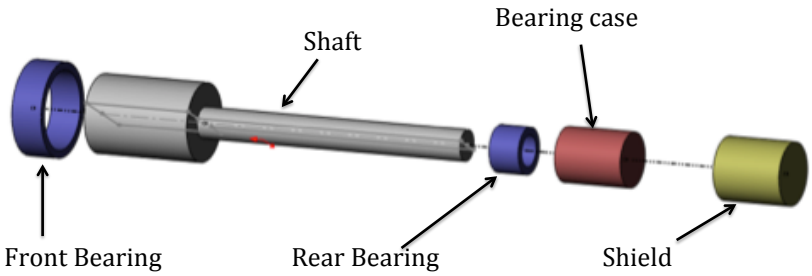


Figure 1: Diagram of the pieces that compose the spindle

Resolution of the two exposed problems are based on different algorithms. The calculation of the optimal nominal dimension of each piece is based on the Hungarian algorithm, while the allocation of the pieces together is treated as a maximum flow problem.

For the first problem a simulation is needed. The pieces have not been produced yet and therefore it is necessary to simulate them.

As expected, the project focuses not only on the theoretical resolution of the problem, but it also develops and explains a practical implementation of it through a computer software. This is the tool that will help the company face the problem and get a real solution for it.

As it is being explained along the project's description, the way in which the manufacturing process is carried out in the company is not as good as it should be. This means that, if these problems are solved correctly, there is a great scope for improvement in many ways: the time spent by the worker at allocating pieces could be reduced, significant savings in the economic aspect could arise, improvements in the quality of the spindles manufactured would take place...

The results obtained after the development of the project are the ones expected. The procedure developed can solve both problems, allowing the calculation of the nominal dimensions of the pieces to produce according to the lots received and then assigning the pieces together.

This enables a better procurement planning and production process, in addition to an improvement in the number of spindles properly produced.

1.3 Motivation

Nowadays performing some tasks through a computer software can involve huge improvements for a company. Often the computer can solve problems in a way that a person is not able to for different reasons: because of the complexity of calculations, because of the amount of possible combinations, due to the necessity of carrying out a simulation...

In a company dedicated to the assembly of parts, where no tool is available to carry out the assignment of them, this task is totally random. A task solved haphazardly rarely allows achieving the best result.

The availability of a tool that allows to automatically assign these pieces together is a great improvement since it allows to replace that random task. In addition to a large decrease in the time spent by the employee to perform the task, it ensures the maximum utilization of available parts and therefore, the production of the largest number of final pieces.

1.4 Aim of the project

The project's aim is the development of a procedure that allows solving the two major problems that the company faces when producing and assembling the pieces that make up the spindle.

The actions that should be performed to solve these problems are:

- Calculate the optimal nominal dimension in the production of parts. When producing the spindle, bearings are bought and the other parts are manufactured. To use the greatest number of bearings that are available, the other pieces must be manufactured based on the dimensions of the bearings.

- Once all batches of parts are available and their dimensions are known, they should be assigned together so that the greatest number of spindles produced is obtained.

The solution of these problems should not only be theoretical, since this would not be useful for the company. It is equally important to develop a tool that allows them to solve the problems practically and to apply it at the factory easily.

1.5 Outline

Before detailing the study and solution of the problems, the necessary theoretical bases are laid for understanding the project. The Hungarian algorithm and the maximum flow problem, the two algorithms used to solve the problems that are studied, are particularly explained.

Then the project is explained in detail. It begins by explaining the product being produced and the process currently followed, as well as the requirements that must be met.

It goes on by explaining the mathematical modeling necessary for solv-

ing the problems, and the strategical approach adopted for this. Once the problems are solved, the solution through a software developed and programmed in Matlab is implemented. The code required for this purpose as well as the elaborated interface is shown.

It then proceeds to the study of the results obtained in order to verify that it has been achieved what was intended at first. The results obtained by assigning parts without calculating their nominal dimension and by calculating them are compared.

To complete the project, some conclusions that summarized the achievements are made as well as some possible improvements in the model itself.

Chapter 2

Theoretical basis

In this section what is sought is to lay the theoretical bases necessary for the understanding of the project, which is presented in the next section. The concepts on which the project is based are discussed. The following content has been obtained from the article of "The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs" , some thesis like "Advancements on problems involving maximum flows", "An Approach to Efficient Network Flow Algorithm for Solving Maximum Flow Problem" or "Flow Optimization for concurrent Multimedia Traffic in Software Defined Networks" and some webpages. All of them are collected in the bibliography at the end of the document.

2.1 Bipartite Matching

A Bipartite Graph $G = (V, E)$ is a graph in which the vertex set V can be divided into two disjoint subsets X and Y, Figure 2, such that every edge $e \in E$ has one end point in X and the other end point in Y. A matching M is a subset of edges such that each node in V appears in at most one edge in M.

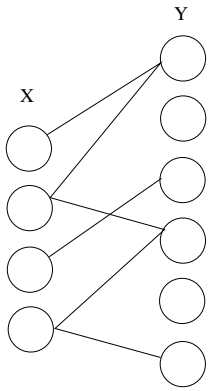


Figure 2: A bipartite graph

What is interesting are matchings of large size. Formally, maximal and maximum matchings are defined as follows:

Maximal Matching: "A maximal matching is a matching to which no more edges can be added without increasing the degree of one of the nodes to two; it is a local maximum"

Maximum Matching: "A maximum matching is a matching with the largest possible number of edges; it is globally optimal"

The goal is to find the maximum matching in a graph. Note that a maximal matching can be found very easily just keep adding edges to the matching until no more can be added. Moreover, it can be shown that for any maximal matching M, $|M| \geq \left|\frac{1}{2}M*\right|$ where $M*$ is the maximum matching. Therefore it can be easily constructed a "2-approximation" to the maximum matching.

Perfect Matching A perfect matching is a matching in which each node has exactly one edge incident on it. One possible way of finding out if a given bipartite graph has a perfect matching is to use Hall's Theorem, Theorem 1.

Theorem 1. *A Bipartite graph $G(V, E)$ has a Perfect Matching iff for every subset $S \subseteq X$ or $S \subseteq Y$, the size of the neighbors of S is at least as large as S, i.e $|\Gamma(S)| \geq |S|$*

2.2 Assignment Problem

Assignment problems arise in those situations where it is necessary to find an optimal way to assign n objects to m other objects. This would be a case of what has been discussed in the previous section.

Depending on the objective to be optimized, different problems can be obtained: linear, quadratic or higher dimensional assignment problems.

These problems are a well studied topic in combinatorial optimization.

An assignment problem is completely specified by its two components:

- The assignments: represent the underlying combinatorial structure.
- The objective function to be optimized: models "the best posible way"

In the classical assignment problem one has $m = n$ and most of the problems with $m > n$ can be transformed or are strongly related to analogous problems with $m = n$. Therefore, it is considered $m = n$ along this explanation.

Assignment problem is one of the special cases of transportation problems. The goal of the assignment problem is to minimize the cost or time of completing a number of jobs by a number of persons. The formulation of this problem, as a special case of transportation problem as just said, can be represented by treating laborers as sources and the tasks as destinations.

To understand, therefore, the problem of assignment is necessary to have a basic understanding of the transportation problem, which consists of the following:

"The transportation problem is a special type of linear programming where the objective is to minimize the cost of distributing a product from a number of sources or origins to a number of destinations. Because of its special structure the usual simplex method is not suitable for solving transportation problems. These problems require a special method of solution. The origin of a transportation problem is the location from which shipments are despatched. The destination of a transportation problem is the location to which shipments are transported. The unit of transportation cost is the cost of transporting one unit of the consignment from an origin to destination.

In most general form, a transportation problem has a number of origins

and a number of destinations. A certain amount of a particular consignment is available in each origin. Likewise, each destination has a certain requirement. The transportation problem indicates the amount of consignment to be transported from various origins to different destinations so that the total of transportation cost is minimized without violating the availability constraints and the requirement constraints."

According to the number of sources and destinations, assignation problems can be classified as:

- Balanced assignment problem: This is an assignment where the number of persons is equal to the number of jobs.

- Unbalanced assignment problem: This is the case of assignment problem where the number of persons is not equal to the number of jobs. A dummy variable, either for a person or job (as it required) is introduced with zero cost or time to make it a balanced one.

Assignment problems can be formulated with techniques of linear programming and transportation problems. As it has a special structure, it is solved by the special method called Hungarian method. This method of assignment problem was developed by a Hungarian mathematician D. Konig and is therefore known as Hungarian method of assignment problem.

To solve the assignment problem using this method, one should know time of completion or cost of making all the possible assignments. Each assignment problem has a table, persons one wishes to assign are represented in the rows and jobs or tasks to be assigned are expressed in the columns. Cost for each particular assignment is in the numbers in the table. It is referred as cost matrix. Hungarian method is based on the principle that if a constant is added to the elements of cost matrix, the optimum solution of the assignment problem is the same as the original problem. Original cost matrix is reduced to another cost matrix by adding a constant value to the elements of rows and columns of cost matrix where the total completion time or total cost of an assignment is zero. This assignment is also referred as the optimum solution since the optimum solution remains unchanged after the reduction.

The Hungarian algorithm is explained below in a more detailed way to understand it the best way possible, because as it has been said previously is the algorithm, in which one of the problems is based.

2.3 Hungarian method

As it has just been commented, the classical solution to the assignment problem is given by the Hungarian or Kuhn-Munkres algorithm, originally proposed by H. W. Kuhn in 1955 and refined by J. Munkres in 1957. The Hungarian algorithm solves the assignment problem in $O(n^3)$ time, where n is the size of one partition of the bipartite graph. This and other existing algorithms for solving the assignment problem assume the a priori existence of a matrix of edge weights, w_{ij}, or costs, c_{ij}, and the problem is solved with respect to these values.

The Hungarian algorithm assumes the existence of a bipartite graph, $G = \langle V, U, E \rangle$ as illustrated in Figure 3, where V and U are the sets of nodes in each partition of the graph, and E is the set of edges. The edge weights may be stored in a matrix as shown in Fig.4. Missing edges are assumed to have zero weight. The minimization form of the problem assumes a matrix of edge costs, w_{ij}=c_{ij} where $W \geq max(w_{ij})$. Missing edges may be given a large cost ($\geq W$), as illustrated in Figure 5.

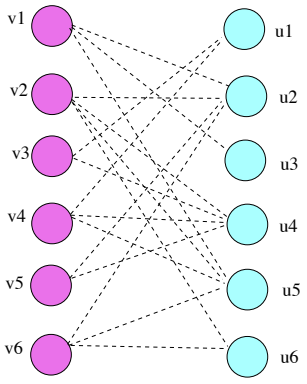


Figure 3: A bipartite graph

w_{ij}	u_1	u_2	u_3	u_4	u_5	u_6
v_1	0	8	9	0	6	0
v_2	0	4	0	5	5	8
v_3	7	0	0	9	0	0
v_4	3	0	0	8	8	0
v_5	0	6	0	7	0	0
v_6	0	8	0	0	9	3

Figure 4: A matrix of edge weights

c_{ij}	u_1	u_2	u_3	u_4	u_5	u_6
v_1	∞	2	1	∞	4	∞
v_2	∞	6	∞	5	5	2
v_3	3	∞	∞	1	∞	∞
v_4	7	∞	∞	2	2	∞
v_5	∞	4	∞	3	∞	∞
v_6	∞	2	∞	∞	1	7

Figure 5: An alternative representation showing edge cost

Each node in the graph may be matched (assigned) or unmatched (unassigned). Unmatched nodes are also called *exposed*. Edges likewise may be matched or unmatched. An edge (v_i,u_j) is matched if v_i is matched to u_j and unmatched otherwise. For clarity, matched edges are designated with solid lines and unmatched edges with dotted lines, as shown in Fig. 6. If v_i

is matched to u_j , u_j is called the mate of v_i, and viceversa. An alternating path is a path through the graph such that each matched edge is followed by an unmatched edge and viceversa. In Fig. 6 (v_5,u_2,v_1,u_1,v_3) is an example of an alternating path. An augmenting path, such as (v_5,u_2,v_1,u_3) in Fig. 6, is an alternating path that begins and ends with an exposed node. All alternating paths originating from a given unmatched node form a Hungarian tree. Searching for an augmenting path in a graph involves exploring these alternating paths in a breadth-first manner, and the process can be called growing a Hungarian tree. Figure 7 illustrates the process of growing a Hungarian tree rooted at node v_5, based on the graph in Figure 6.

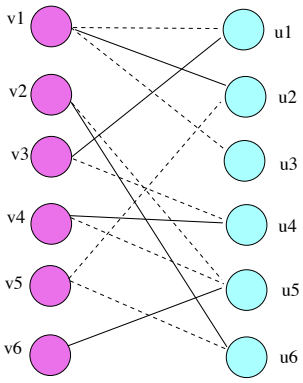


Figure 6: A bipartite graph showing matched and unmatched edges

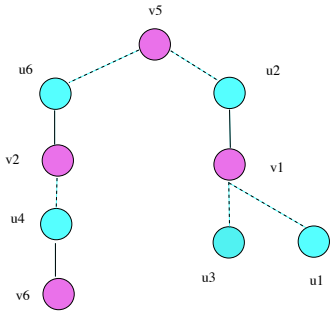


Figure 7: A Hungarian tree rooted at v_5

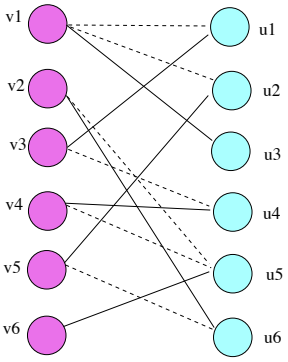


Figure 8: The bipartite graph from Figure 7, with matched and unmatched edges flipped along augmenting path (v_5,u_2,v_1,u_3)

The algorithm assigns dual variables α_i to each node v_i and dual variables β_j to each node u_j. It exploits the fact that the dual of the minimization version of the assignment problem is feasible when $\alpha_i + \beta_j \leq c_{ij}$. The Hungarian algorithm maintains feasible values for all the α_i and β_j from initialization through termination. An edge in the bipartite graph is called admissible when $\alpha_i + \beta_j = c_{ij}$. The subgraph consisting of only the currently admissible edges is called the equality subgraph. Starting with an empty matching, the basic strategy employed by the Hungarian algorithm is to repeatedly search for augmenting paths in the equality subgraph. If an augmenting path is found, the current set of matches is augmented by

flipping the matched and unmatched edges along this path, as illustrated in Fig. 8. Because there is one more unmatched than matched edge, this flipping increases the cardinality of the matching by one, completing a single stage of the algorithm. If an augmenting path is not found, the dual variables are adjusted to bring additional edges into the equality subgraph by making them admissible, and the search continues. n such stages of the algorithm are performed to determine n matches, at which point the algorithm terminates.

If the size of the two partitions of the graph are not equal, a typical strategy is to insert into the relevant partition, dummy nodes with zero-weight edges to all nodes in the opposite partition. As such, the Hungarian algorithm always returns a complete matching, but this matching may include some zero-weight edges, representing "no assignment".

Mathematical explanation

Input: A bipartite graph, $\langle V, U, E \rangle$ (where $|V| = |U| = n$) and an $n \times n$ matrix of edge costs C

Output: A complete matching, M

1. Perform initialization:

   (a) Begin with an empty matching, $M_0 = \emptyset$

   (b) Assign feasible values to the dual variables α_i and β_j as follows:

$$\forall v_i \in V, \alpha_i = 0 \tag{2.1}$$

$$\forall u_j \in V, \beta_j = min(c_{ij})$$

2. Perform n stages of the algorithm, each given by the routine *Stage*.

3. Output the matching after the n^{th} stage: $M = M_n$.

Stage

1. Designate each exposed (unmatched) node in V as the root of a Hungarian tree.

2. Grow the Hungarian trees rooted at the exposed nodes in the equality subgraph. Designate the indices i of nodes v_i encountered in the Hungarian tree by the set $I*$, and the indices j of nodes u_j encountered in the Hungarian tree by the set $J*$. If an augmenting path is found, go to step (4). If not, and the Hungarian trees cannot be grown further, proceed to step (3).

3. Modify the dual variables α and β as follows to add new edges to the equality subgraph. Then go to step (2) to continue the search for an augmenting path.

$$\theta = \frac{1}{2} \min_{i \in I*, j \notin J*} (c_{ij}, \alpha_i, \beta_j) \tag{2.2}$$

$$\alpha_i \leftarrow \begin{cases} \alpha_i + \theta & i \in I* \\ \alpha_i \ \ \theta & i \notin I* \end{cases} \tag{2.3}$$

$$\beta_j \leftarrow \begin{cases} \beta_j + \theta & j \in J* \\ \beta_j \ \ \theta & j \notin J* \end{cases} \tag{2.4}$$

4. Augment the current matching by flipping matched and unmatched edges along the selected augmenting path. That is, M_k (the new matching at stage k) is given by $(M_{k\ \ 1}\ \ P) \cup (P\ \ M_{k\ \ 1})$ where $M_{k\ \ 1}$ is the matching from the previous stage and P is the set of edges on the selected augmenting path.

Note: θ can be efficiently computed in $O(n)$ rather than $O(n^2)$time by maintaining "slack variables" during the search for an augmenting path in step (2). That is, during the search, it is maintained for each node u_j, $slack(u_j) = \min_{i \in I*}(c_{ij}\ \ \alpha_i)$.

Then, $\theta = \frac{1}{2}\min_j(slack(u_j)$

To summarize, the steps to be followed to solve a problem using the Hungarian algorithm are reflected in the Figure 9:

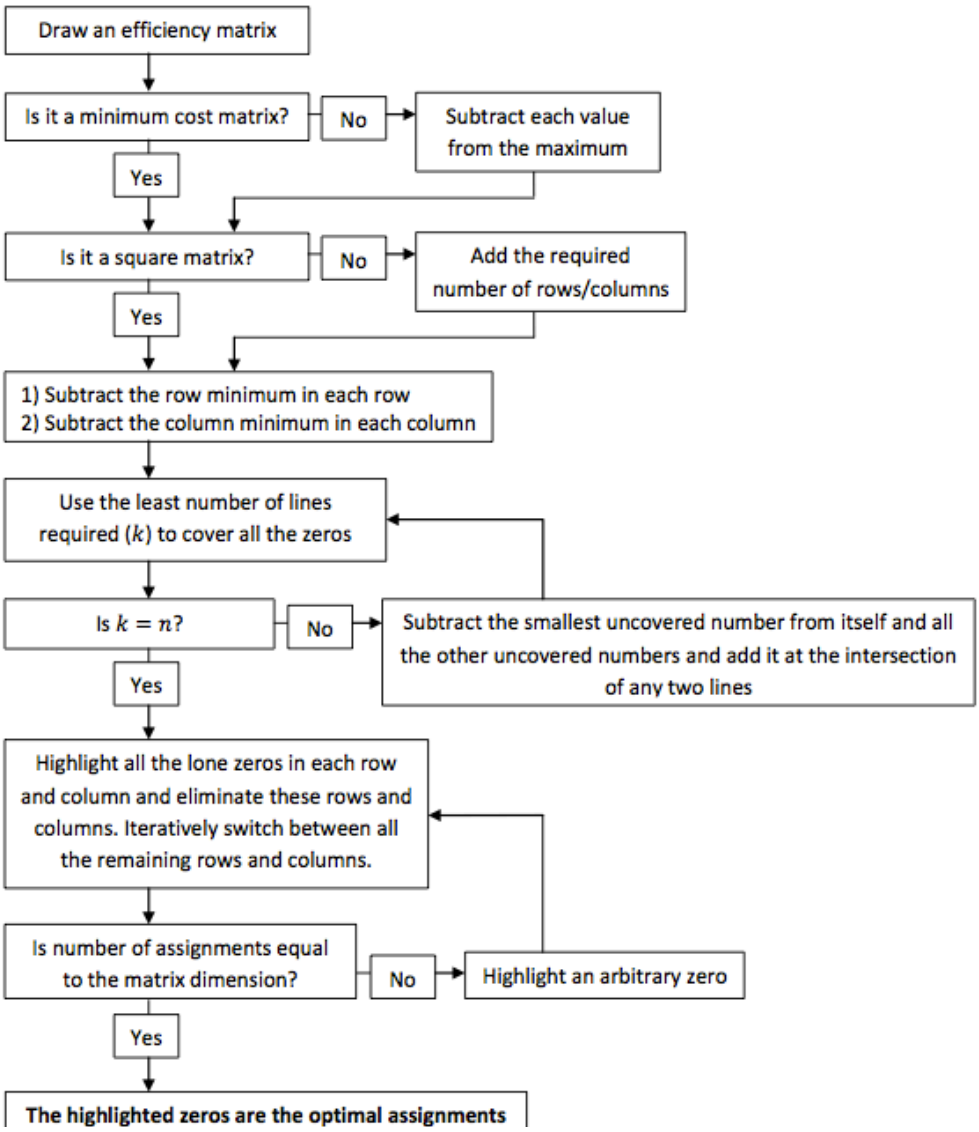


Figure 9: Diagram of how to implement the Hungarian algorithm step by step

2.4 Network flow

Network flows typically refers to the study of algorithms to solve optimization problems on networks.

The common abstraction that models a flow network is a directed graph G = (V, E), which consists of a set V containing the vertices or nodes and a set E containing the edges. Each edge $e \in E$ is a tuple of two vertices e = (u, v) with $u, v \in V$. The order of the graph is equal to the cardinality of the set of vertices $|V|$. The size of the graph is equal to the cardinality of the set of edges $|E|$.

Generally speaking, a network is a discrete structure that defines connectivity relationships amongst a set of nodes. In this context, an arc denotes a direct connection between two nodes. Arcs typically have an orientation, which indicates which node the arc originates from and which node the arc terminates in.

There are two types of graphs depending on its orientation: a graph is called directed if its edges have a defined orientation, else undirected. In case of the directed graphs this means $(u, v) \neq (v, u)$, whereas in the undirected case $(u, v) = (v, u)$ holds.

A flow on an arc denotes the number of units of the commodity in question, be it electricity, freight, fluid, et cetera, that will traverse the arc. A special source vertex s, that belongs to V, produces these units that flow through the edges of the graph to be consumed by a sink vertex t, that belongs to V. That means that s, the source vertex, has no incoming edge, and t, the sink vertex, has no outgoing edge. A network flow is an assignment of a value of flow to each arc in the network. A network flow is balanced if, for each node, the total amount of the flow entering the node equals the total amount of the flow leaving the node. In addition, arcs have flow bounds, that respectively denote the minimum and maximum amount of flow that may traverse an arc. Often, an upper bound on the amount of flow that may traverse an arc is called an arc capacity.

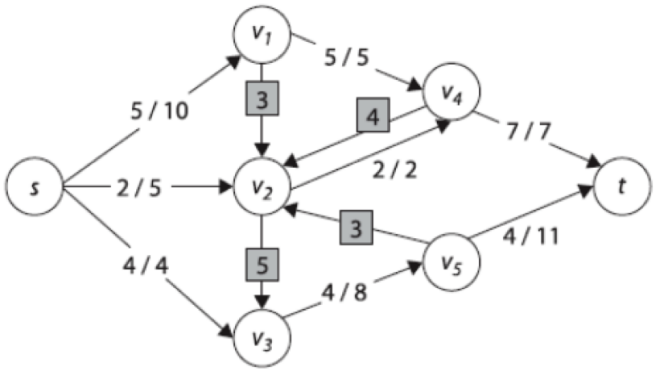


Figure 10: A maxflow problem's diagram

The image in Figure 10 shows an example of this type of graph. Each edge (u, v) has a flow $f(u, v)$ that defines the number of units that flows from u to v. An edge also has a capacity $c(u, v)$ that constrains the maximum number of units that can flow over that edge. In Figure 10, each vertex is numbered (with vertices s and t clearly marked) and each edge is labeled as f/c, showing the flow over that edge and the maximum possible flow. The edge between s and v1, for example, is labeled 5/10, meaning that 5 units flow over that edge, which can sustain a capacity of up to 10. When no units are flowing over an edge (as is the case with the edge between v5 and v2), only the capacity is shown, outlined in a gray box.

A formal definition of a flow network is given by:

Let $G = (V, E)$ be a finite, directed graph with positive weights w(u,v) and capacities $c(u, v)$ where $(u, v) \in E$. A flow in this network is given by a function $f : VxV \rightarrow \mathbb{R}$ for which the following properties hold:

- the *capacity limit* is not violated for any $(u, v) \in E$: $f(u, v) \leq c(u, v)$.

- The *flow balance* $\sum_{(u,v)\in E^-(u)} f(u, v) = \sum_{(u,v)\in E^+(u)} f(v, u)$ holds for all $(u, v) \in E$. E^- is the set of outgoings edges with respect to u and E^+ is the set of incoming edges with respect to u. An exception for this rule are the source node s and the sink node t. At a source typically a positive amount of flow is present and at a sink typically a negative amount of flow is present. In short this menas that each unit of flow entering a node has to leave a node again except for s and t.

- The *flow preservation* $\sum_{(u,v)\in E} f(u,v) = 0$ holds for the complete network. This means that no amount of flow is lost in the network.

- The *skew symmetry* property for the flow function: $\forall(u,v) \in E : f(u,v) = -f(v,u)$. In other words, only a flow which has already passed an edge can pass it in the opposite direction.

In the ensuing algorithms a *network path* is a non-cyclic path of unique vertices $< v_1, v_2, ..., v_n >$ involving $n-1$ consecutive edges (v_i, v_j) in E. In the directed graph shown in Figure 10, one possible network path is $< v_3, v_5, v_2, v_4 >$. In a network path, the direction of the edges can be ignored.

The value of a flow f is defined as $|f| = \sum_{v\in V} f(u,v) = 0$. That is, the total flow out of the source. Here, the $|\cdot|$ notation denotes flow value, not absolute value or cardinality.

2.4.1 Maximum Flow Problem

This problem can be stated as follows: given a network with a source, a sink and a capacity on each arc, find the maximum possible flow that can be routed through the network from the source to the sink. This problem is often used to model real-world situations where the routing costs are negligible relative to the profit per unit of flow.

There are many interesting properties of the MFP. First of all, there is the property of integrality. If the capacities on all of the arcs in the network are integers, then there will be a maximum flow in the network such that every arc will assume an integer value and the total flow through the network will be integer. This property is important because integrality constraints are typically very difficult to enforce in practice, which is why integer linear programs are much more difficult to solve in practice than linear programs. The integrality property of maximum flows allows the integrality on the arc flows to be guaranteed by merely assigning integer arc capacities. Thus, a maximum flow problem can be solved as a linear program, despite integrality requirements on the flow.

The maximum flow problem is a classical optimization problem with many applications. The problem of finding a maximum flow in a directed graph with edge capacities arises in many settings in operations research

and other fields, and efficient algorithms for this problem have been studied for over four decades. In this problem, the greatest rate at which material can be shipped from the source to the sink without violating any capacity constraints is calculated. It is one of the simplest problems concerning flow networks and this problem can be solved by efficient algorithms. Moreover, the basic techniques used in maximum-flow algorithms can be adapted to solve other network-flow problems.

A maximum-flow problem may have several sources and sinks, rather than just one of each. However, the problem of determining a maximum flow in a network with multiple sources and multiple sinks is reduced to an ordinary maximum-flow problem.

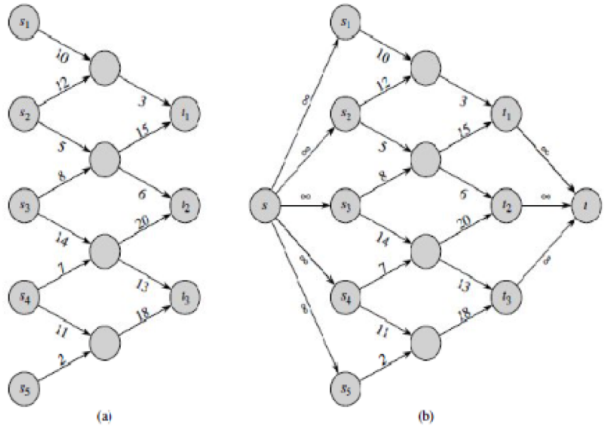


Figure 11: Converting a multiple-source, multiple-sink maximum-flow problem into a Problem with a single source and a single sink. (a) A flow network with five sources $S = s_1, s_2, s_3, s_4, s_5$ and three sinks $T = t_1, t_2, t_3$. (b) An equivalent single- source, single-sink flow network.

Figure 11 (b) shows how the network from (a) can be converted to an ordinary flow network with only a single source and a single sink. A supersource s and a directed edge (s, s_i) is added with capacity $c(s, s_i) = \infty$for each $i = 1, 2, ..., m$. A new supersink t and a directed edge (t_i, t) with capacity $c(t_i, t) = \infty$ for each $i = 1, 2, ..., n$. Intuitively, any flow in the network in (a) corresponds to a flow in the network in (b), and vice versa.

The single source s simply provides as much flow as desired for the

multiple sources s_i, and single sink t likewise consumes as much flow as desired for the multiple sinks t_i.

Since this problem is a network flow problem, the properties that define this network flow must be satisfied. These properties are:

- Capacity constraint: For all $u, v \in V$, it required that $f(u, v) \leq c(u, v)$.
- Flow conservation: For all $u \in V - s, t$, it required that $\sum_{v \in V} f(u, v) = 0$
- Skew symmetry: For all $u, v \in V$, it required that $f(u, v) =? f(v, u)$.

The network associated with this type of problem is denoted by $R = [X, A, q]$ where:

X represents the group of nodes. It's assumed that the cardinality of X is n.

A represents the set of paths (arcs) between nodes among which the flow can be transportd ie. $A = \{(1, j)/i, j \in X\}$, the product can be sent from i to j. The cardinality of A is denoted by m.

q is a function $q : A \to Z$ where each $(i, j) \in A$ associate the maximum transport capacity from i to j. q is called capacity function.

By v the quantity of flow on the network from node s to node t is denoted and by f_{ij} the amount of flow from node i to node j. Our problem then is:

Maximize v

subject to:

$$\sum_{j \in \Gamma^+(i)} f_{ij} - \sum_{k \in \Gamma^-(i)} f_{ki} = \begin{cases} v & \text{if } i = s \\ 0 & \text{if } i \neq (s) \ \ or \ \ i \neq (t) \\ -v & \text{if } i = t \end{cases} \tag{2.5}$$

$$0 \leq f_{ij} \leq q(i, j) \quad \forall (i, j) \in A \tag{2.6}$$

(3.17) is known as *flow conservation equations* and their left side is the amount of flow leaving minus the amount of flow into each node, which takes the values shown on the right. (3.16) means that the flow of each arc must be less or equal to the maximum allowable flow for him.

A feasible flow in a network is defined as follows:.

Definition: $R = [X, A, q]$ is a function $f : A \rightarrow Z$ that satisfies the conditions 1 and 2 above written. It is said that f is maximum when generating the highest possible value of v. Given a graph $G = (V, E)$ the adjacency-matrix A is a $|V|$ x $|V|$ matrix $A = (a_{ij})$ with

$$a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & otherwise \end{cases} \tag{2.7}$$

2.5 Data analysis

A common kind of statistical inference is hypothesis testing. Statistical data analysis allows to use mathematical principles to decide how likely it is that the sample results match the hypothesis about a population.

2.5.1 ANOVA

Analysis of variance (ANOVA) is a statistical method used to interpret experimental data and make necessary decisions. It detects any differences in the average performance of groups of items tested.

To carry out this analysis a null hypothesis (H_0) (2.8) and an alternative hypothesis (H_1) (2.9) are needed. The null hypothesis is the assumption that there will be no differences between groups that are tested, or what is the same, that population's means for all groups are the same. The alternative hypothesis, on the other hand, is the hypothesis stating that there will be a difference between groups.

Mathematically it would be expressed like:

$$H_0 : \quad \mu_1 = \mu_2 = \ldots = \mu_k = \mu \tag{2.8}$$

$$H_1 : \quad \exists \mu_j \neq \mu \quad j = 1, 2, \ldots, k \tag{2.9}$$

In general, one way ANOVA techniques can be used to study the effect of $k > 2$ levels of a single factor.

When applying one way analysis of variance there are three key assumptions that should be satisfied:

1. The observations are obtained independently and randomly from the populations defined by the factor levels.

2. The population at each factor level is (approximately) normally distributed.

3. These normal populations have common variance, σ^2.

Thus for factor level i, the population is assumed to have a distribution which is $N(\mu_i, \sigma^2)$

Once the ANOVA is performed, an analysis of the results is necessary, which can be carried out through the p-value and the graphical support of a box-plot.

- p-value
  In statistical hypothesis testing it is used a p-value (probability value) to decide whether or not the sample provides strong evidence against the null hypothesis. The p-value is a numerical measure of the statistical significance of a hypothesis test. It tells how likely it is that the sample data could have been gotten even if the null hypothesis is true. By convention, if the p-value is less than 5% ($p < 0.05$), it is concluded that the null hypothesis can be rejected. In other words, when $p < 0.05$ it is said that the results are statistically significant, meaning there is a strong evidence to suggest the null hypothesis is false.
  If the null hypothesis is rejected, then all that it is known is that at least 2 groups are different from each other. In order to determine which groups are different from which, post-hoc t-tests are performed using some form of correction to adjust for an inflated probability of a Type I error.

- Box-plot
  A box-plot is a graphical data analysis technique for visualizing if differences exist between the various levels of a 1-factor model. The box-plot is a graphical complement to 1-factor ANOVA. It is also a useful technique for summarizing and comparing data from 2 or more samples.
  The box-plot usually looks like the one in Figure 12

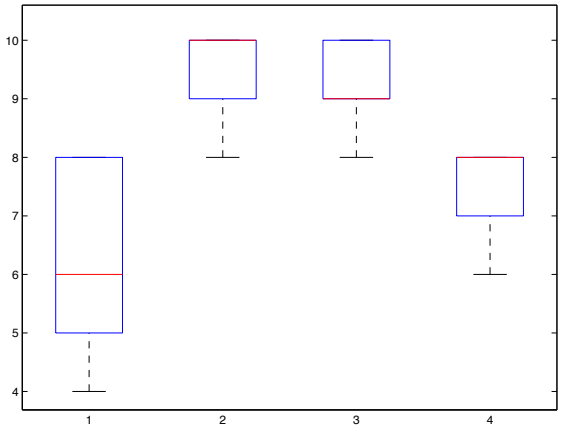


Figure 12: A box plot

This allows therefore to perform a visual analysis to compare the results for each group that is simulated.

Chapter 3

Project

3.1 Description of the problem

3.1.1 Description of the products

The product in which the project focuses is the spindle produced by the company.

PRECISE Technologies specialises in spindles with integrated high-frequency motors using synchronous or asynchronous technology, and delivering speeds of up to 160,000 rotations per minute (rpm). PRECISE spindles are designed for high-precision fabrication with tolerances of less than a micrometre, and represent the ideal solution for precision milling, drilling, and grinding applications in which small spindles are indispensible.

This spindle, as already said, is composed of many elements. However, just some of them are taken into account, particularly those that produce more problems in the production process.

These elements are:

- Shaft : around it are placed the other elements. Only two diameters of it are important and they are those of the extremes. They are referred as front and rear shaft.

- Two ball-bearing : one of them is located in the front of the shaft and the other one is in the back. What is important is the inside diameter, which is related to the shaft, and the outer diameter, which is related to the bearing case. These pieces are those that the company purchases and not manufacture.

- Bearing case: it is related to the rear bearing and the housing. As with the rear bearing both diameters, inner and outer, are important. The inner one is related to the rear bearing and the outer one with the housing.

- Shield/Housing: it is related to the bearing case, more specifically with its outside diameter. What is important is the inside diameter.

From these elements not only the diameters that have been named are important, but also the relationship between them.

In the Figure 13 the arrangement of the elements of the spindle and the diameters needed for the project can be seen. They are the ones that have just been named.

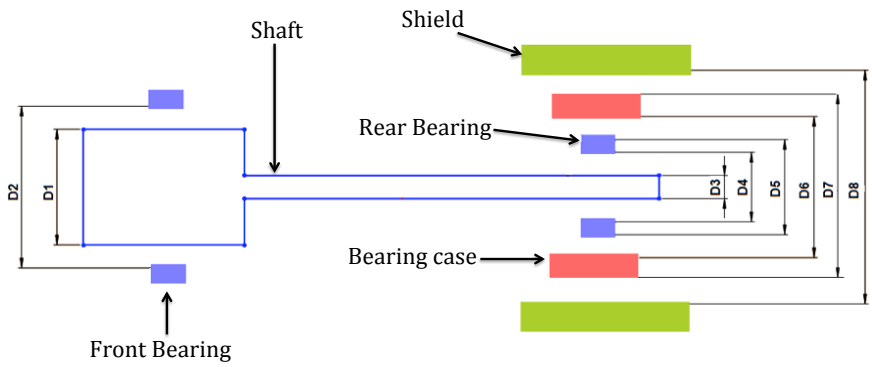


Figure 13: Spindle's diameters on which the project focuses

In Figure 13 the shaft is represented by the white piece; the two bearings are the purple ones, the red one is the bearing case and the green one is the shield.

As it has been explained and as can be seen in Figure 13, 8 diameters are important. These diameters are:

- Shaft
  - Front shaft's diameter: D1
  - Rear shaft's diameter: D3
- Front Ball-bearing
  - Internal diameter: D2
- Rear Ball-bearing
  - Internal diameter: D4
  - External diameter: D5
- Bearing case
  - Internal diameter of the bearing case: D6
  - External diameter of the bearing case: D7
- Shield
  - Internal diameter of the shield: D8

These 8 diameters must meet 4 certain relationships between them for a correct operation of the spindle.

These relationships are:

1) The relationship between the front shaft's diameter and the inner diameter of the front bearing (D1-D2). Between the front shaft's diameter (D1) and the inner diameter of the front bearing (D2) there is a press fit. According to what is stated in the assembly drawings, this setting is valid whenever it is between 3 and 5 μm. Since it is a press fit, the shaft should be larger than the bearing for this to be true.

2) The relationship between the rear shaft's diameter and the internal diameter of the rear bearing (D3-D4). As in the previous case, between the rear shaft's diameter (D3) and the internal diameter of the rear bearing (D4) exists a press fit and it is valid when it is between 3 and 5 μm. Everything that has been set out in paragraph 1 is applicable to this case.

3) The relationship between the outer diameter of the rear bearing and the inner of the bearing case (D5-D6). Between the outer diameter of the rear bearing (D5) and the inner one of the bearing case (D6) there is a loose fit. According to what is stated in the assembly drawings, this setting is valid whenever it is between 6 and 8 μm. Since it is a loose fit, the bearing case should be larger than the bearing for this to be true.

4) The relationship between the outer diameter of the bearing case and the inner of the shield. (D7-D8). As in the previous case, between the outer diameter of the bearing case (D7) and the internal diameter of the shield (D8) exists a loose fit but in this case, it is valid when it is between 1 and 2 μm. Since it is a loose fit, the shield should be larger than the bearing case for this to be true.

However the calculation of the nominal dimensions shall be carried out only to those combinations in which the bearings is involved. In combination 4 both parts are manufactured by the company and this is a more complex situation, as it would depend on which of the two elements is made before the other and they would have to wait till one was manufactured to produce the other, which would delay the production of the spindle.

These relationships between parts are included in the manufacture's and assembly's drawings, which must be known by the employees.

Now, all possible combinations of measures that can be found in the cases just exposed are collected in some tables.

In the first table, Figure 14, the allowable dimensions of the shafts (in mm) that are produced as well as the allowable dimensions of the purchased bearings (in mm) are collected. Those parts that are not within these dimensions are not valid.

For the front bearings to be admissible they must measure between 25 and 24.997 mm, while the shafts produced must always measure between 25.001 and 25.004 mm. All measurements out of these values are not considered.

Any manufactured part that does not meet with these limits would be submitted again to treatment. If the measure is below the range, material is added to the piece and then it is again subjected to the appropriate treatment to remove material until it is within the range. If, on the contrary, it is above the range, material is removed directly until a valid dimension is obtained.

As it has been said, between the front shaft and the front bearing there is a press fit, whose value must be between 5 and 3 μm., since otherwise the spindle does not work properly.

According to what has been just explained, the Figure 14 can be drawn:

		Int. Front bearing			
		25,000	24,999	24,998	24,997
front shaft	25,001	-0,001	-0,002	-0,003	-0,004
	25,002	-0,002	-0,003	-0,004	-0,005
	25,003	-0,003	-0,004	-0,005	-0,006
	25,004	-0,004	-0,005	-0,006	-0,007

Figure 14: Possible relationships between front shaft's and bearing's diameters

Cells in green represent the combination that may be admissible, while cells in red represent those that are not acceptable. These are those combination with a result greater than 5 or lower than 3 μm.

Those combinations that appear in red cells can never be obtained as a solution when the problem is solved.

This table could be performed for any of the cases, since it is just needed to know the allowable dimensions of each piece and the acceptable tolerances of the Spindle.

Below the tables with these combinations in the case of the other elements that compose the spindle are collected.

In the case of the rear shaft and the bearing, Figure 15, tolerance to be met is the same as with the front shaft just discussed.

		Int. Rear bearing			
		22,000	21,999	21,998	21,997
rear shaft	22,001	-0,001	-0,002	-0,003	-0,004
	22,002	-0,002	-0,003	-0,004	-0,005
	22,003	-0,003	-0,004	-0,005	-0,006
	22,004	-0,004	-0,005	-0,006	-0,007

Figure 15: Possible relationships between rear shaft's and bearing's diameters

Both in the case of bearing case and shield these relationships change.

Regarding to the bearing case, Figure 16 the assignment is valid when the subtraction of the diameters is between 6 and 8 μm.

		Ext. Rear bearing			
		42,000	41,999	41,998	41,997
Int. Bearing case	42,006	0,006	0,007	0,008	0,009
	42,007	0,007	0,008	0,009	0,010
	42,008	0,008	0,009	0,010	0,011

Figure 16: Possible relationships between bearing cases' and bearing's diameters

In relation to the housing, Figure 17 the assignment is valid when the subtraction of the diameters is between 1 and 2 μm.

		Ext. Bearing case		
		60,001	60,000	59,999
Housing	60,000	-0,001	0,000	0,001
	60,001	0,000	0,001	0,002
	60,002	0,001	0,002	0,003
	60,003	0,002	0,003	0,004

Figure 17: Possible relationships between housing's and bearing cases' diameters

Again the green cells show the results that would be accepted, while the ones in red show those that should be discarded and never accepted as a solution.

3.1.2 Description of the current company's procedure to produce and assembly the parts

Having made the explanation of the product, for which the problem has to be solved, the current procedure carried out in the company to produce the

spindles is explained.

Nowadays all the elements of the spindle but the ball-bearings, which are bought in batches, are produced. The number of pieces making up these batches depend on the number of spindles to be manufactured.

Although specific dimensions for bearings are requested, they can have a certain deviation, which must be taken into account when to manufacture and to fit the elements. The existence of these deviations is expected and does not have to be a major problem for the company as long as it is known how to handle it properly.

Because of these deviations, the other elements that make up the spindle must be produced in function to the received bearings in each batch and their measurements.

In order to obtain that the largest number of parts manufactured and bought fit, these are manufactured searching an intermediate tolerance. However, this does not ensure that the greatest number of combinations is reached, since, as just explained, this depends on the bearings received.

The ideal situation would be that workers know in advance the nominal measurement of each part to be produced as a function of each batch of bearings received.

Moreover there is a lack of a method to assign the pieces together so it's not possible to obtain the largest possible number of optimal products. What is currently done to assign the elements together is to take one of them and try to fit it into other. If it fits, it is let that way; if it doesn't, another element is taken and the action is repeated till one fits.

Since this procedure is totally random, it's not an appropriate one. The number of bearings that must be purchased to finish a certain number of spindles is really large, sometimes more than doubled.

This procedure involves, therefore, a loss of profit for the company, both economic and time. Many more bearings than the ones needed are purchased, the worker must devote much time to the process of fitting the pieces and also it's not obtained as many optimal spindles as it could be obtained.

This means that if a suitable procedure is achieved, significant improve-

ments for the company can be obtained.

What is sought, in short, is an improvement on the way to carry out these processes so as to obtain a benefit, not necessarily an economic one.

3.1.3 Main goals of the optimization

As it has been described in the previous paragraph, some improvements can be carried out in the production process of the spindle.

These improvements can make important business benefits, both economic and related to time invested. It can mean a reduction in the number of bearings that must be purchased, an increase in the number of optimal spindle produced, a reduction in the time required by the worker to fit the pieces...

The aim of this project is to find a mathematical or theoretical solution that allows the optimization of the production process currently performed. To achieve this it is necessary to:

- obtain the nominal dimensions of the different elements to be produced depending on the bearings that have been purchased. This problem is solved by a simulation using the Hungarian algorithm.

- obtain the best possible combination between all the available elements so that the greatest number of optimum spindle occurs. This problem is solved as a maximum flow problem.

The project, however, does not end here because, although the problem is solved in a mathematical way, this is not useful for the company. The easiest way for the company to apply this developed solution should also be found. Otherwise they could not bring solution to their problem.

3.2 Mathematical modeling

The optimization is the selection of a better alternative, in some sense, as other possible alternatives. It is a concept inherent in all operations research. However, certain characteristics of operations research techniques are collected under the name of optimization and mathematical programming.

Optimization problems are generally composed of:

- Objective function:
  It is the quantitative measure of system's performance to be optimized (maximize or minimize). As an example of objective functions can be mentioned: the minimization of variable costs of operating a power system, maximizing the net benefits of selling certain products, minimization of the squared deviations about values, the minimization the material used in the manufacture of a product, etc.

- Variables:
  They represent the decisions that can be taken to affect the value of the objective function. From a functional point of view can be classified into principal or independent variables and auxiliary or dependent or state variables, although mathematically are all the same. In the case of sale, they will be the amount of each product produced and sold. For the manufacture of a product, its physical dimensions.

- Restrictions:
  They represent the set of relationships (expressed by equations and inequalities) that certain variables are required to meet. For example, the maximum and minimum operating power generating group, the production capacity of the factory for different products, the dimensions of the raw materials of the product, etc.

Solving an optimization problem involves finding the value to be taken by the variables to make the objective function optimal satisfying the set of constraints.

Optimization methods can be classified into: classical methods (which are the algorithms that usually are explained in the books of optimization) and methaheuristic methods (which were linked to what artificial intelligence was called and imitate simple phenomena observed in nature).

Within the first methods are linear optimization, linear mixed integer nonlinear stochastic dynamics, etc. In the second group are included evolutionary algorithms (genetic and others), the method of simulated annealing (simulated annealing), heuristic searches (method taboo, random search, greedy, etc.) or multi-agent systems.

In a very general and roughly way, it can be said that the classical methods seek and ensure a local optimum while methaheuristic methods

have specific mechanisms to achieve a global optimum but do not guarantee that it's scope.

3.2.1 Mathematical modeling of the nominal dimension's determination

Focusing on our problem, the calculation of nominal dimensions is solved by applying a linear optimization that responds to the following structure of linear programming:

$$\min_x(c^T x)$$

$$Ax = b$$

$$x \geq 0$$

$$x \in \mathbb{R}^n, \quad c \in \mathbb{R}^n, \quad A \in \mathbb{R}^{m \times n}, \quad b \in \mathbb{R}^m$$

Specifically applied to the problem:

$$f(x) = \sum_i^m \sum_j^n c_{ij} x_{ij} \tag{3.1}$$

What is sought by applying the Hungarian algorithm previously explained, is to obtain the smallest possible value for the function $f(x)$, i.e. :

$$\min_{i>0, j>0} = \sum_i^m \sum_j^n c_{ij} x_{ij} \tag{3.2}$$

Particularly, on the problem being solved these variables are defined as:

c_{ij} are the elements of the cost matrix, whose obtainment is explained below.

As far as the variable x_{ij} is concerned, there are only two possible values for it:

$$x_{ij} = \begin{cases} 1 & \text{if combination of bearing } i \text{ and element } j \text{ is a solution of the problem} \\ 0 & \text{if the combination is no solution} \end{cases} \tag{3.3}$$

The faced problem has many restrictions. First, there are those that define the allocation problem and that have been already explained in the section of the theoretical explanation. These restrictions are:

$$\sum_j x_{ij} \leq 1, \quad i = 1, 2, ..., n \quad \textit{where } n \textit{ is the number of bearings} \tag{3.4}$$

$$\sum_i x_{ij} \leq 1, \quad j = 1, 2, ..., m \quad \textit{where } m \textit{ is the number of elements} \tag{3.5}$$

As far as this problem is concerned, the first restriction (3.4) means that each bearing i can not be assigned to more than one element j (front shaft, rear shaft or bearing case), while the second (3.5) means that each element j can not be assigned to more than one bearing i.

Moreover, the problem is affected by other restrictions, such as those determined by the dimensions of the parts and the relations between elements that must be met for the proper functioning of the final spindle. Constraints that must be met are, therefore:

$$25,001 \leq D_1 \leq 25,004$$

$$24,997 \leq D_2 \leq 25$$

$$22,001 \leq D_3 \leq 22,004$$

$$21,997 \leq D_4 \leq 22$$

$$41,997 \leq D_5 \leq 42$$

$$42,006 \leq D_6 \leq 42,008$$

$$59,999 \leq D_7 \leq 60,001$$

$$62,000 \leq D_8 \leq 62,003$$

$$0,003 \leq D_1 - D_2 \leq 0,005$$

$$0,003 \leq D_3 - D_4 \leq 0,005$$

$$0,006 \leq D_6 - D_5 \leq 0,008$$

$$0,001 \leq D_8 - D_7 \leq 0,002$$

Now the mathematical problem just discussed is explained. As it has been said, the program must solve three possible cases, depending on the diameter to be calculated. In all these cases the mathematical modeling is the same: regarding to the objective function, j refers to the elements of which the dimensions has to be calculated and later assigned (front shaft, rear shaft or bearing case); i refers to the bearings.

First thing to be done is to calculate the difference between the dimension of the piece j the dimension of the bearing i. What is got after this step is a matrix like Fig. 14-Fig. 17.

However, that's not the matrix to be used. A transformation of the matrix should be done in order to obtain another matrix on which the Hungarian algorithm is applied.

What is sought is the minimization of the objective function (3.2) (that is what the Hungarian algorithm handles), so all acceptable combinations have to be replaced for a "0" and those that can not be accepted by an "infinite value".

Therefore a matrix composed only by 0 and infinite values is obtained. That's the cost matrix and each element making up it is c_{ij} of the function 3.2 previously exposed.

The assignment problem that should be solved in this research differs from the original setting the Hungarian algorithm was designed for in two distinct ways. Firstly, the algorithm was developed to assign n "persons" to n "jobs". Another discrepancy is the fact that the algorithm gives a solution in terms of minimum cost, whereas this problem doesn't work with the cost of the pieces, but with the relationships between their diameters.

The practical resolution of this algorithm has already been explained in paragraph 2, the one of the theoretical explanation.

It must be said that in this case, the resolution of this algorithm does not take place in a practical way, as it would mean a useless and very laborious effort. Instead, it is held through a calculation tool such as Matlab.

In the calculation of the nominal dimensions the resolution of this algorithm shall be carried out on numerous occasions as this calculation is based on a simulation. This means that this process is to be followed for each simulation.

In this case, after solving the above problem and finding the combinations of parts that minimizes the objective function in each simulation, it should be chosen which of these solutions is the best one.

However this is not complicated. The result obtained in each simulation is the number of combinations that can be got. Therefore what has to be done is to find the corresponding mean value of each simulated dimension, just by adding the results of each simulation and dividing it by the number of simulations carried out.

The dimension, for which a higher value is obtained, is the nominal dimension that should be produced.

In the next section "strategical approach" this modeling is explained through a sample, allowing to understand it better.

3.2.2 Variability of the parts' dimension in the production process

According to what it has just been exposed, an aspect that requires an explanation is the simulation of the parts. In order to carry out this simulation, just the dimension to be simulated and the deviation are needed.

To calculate the deviation in each production process the following procedure should be carried out.

X is the characteristic of quality that is wanted to be measured, in this case the diameter of the piece, where $X \approx N(\mu, \sigma)$. They are taken k samples each of size n. It is denoted by $x_{i1}, x_{i2}, \ldots x_{in}$ to the n observations forming the i^{th} sample, where $i = 1, \ldots, k$.

If μ is unknown, it can be estimated (note that this calculation should be based on the samples obtained k taken when it is considered that the process is under control):

$$\hat{\mu} = \bar{\bar{X}} = \frac{1}{k} \sum_{i=1}^{k} \bar{X}_i \tag{3.6}$$

where $\bar{X}_i = \frac{1}{n} \sum_{j=1}^{k} X_{ij}$

Note that $\hat{\mu}$ is an unbiased estimator since $E[\bar{\bar{X}}] = \frac{1}{k} \sum_{i=1}^{k} E[\bar{X}_i] = \mu_x = \mu$.

If σ is unknown, it can be estimated from the standard deviations S_i (as in the case of μ, note that this calculation should be based on the samples obtained k taken when it is considered that the process is under control): $\forall 1, 2, \ldots, k$ is

$$S_i = \sqrt{\frac{\sum_{j=1}^{n}(X_{ij} - \bar{X}_i)^2}{n-1}} \tag{3.7}$$

It's true that $\mu_{Si} = c_4(n) \times \sigma$ where $c_4(n)$ is a weighted value dependent on n.

Specifically, $c_4(n)$ is:

$$c_4(n) = (\frac{2}{n-1})^{1/2} \frac{\Gamma(n/2)}{\Gamma[(n-1)/2]} \tag{3.8}$$

Note that $S_i/c_4(n)$ is an unbiased estimator of σ because:

$$E[\frac{S_i}{c_4(n)}] = \frac{E[S_i]}{c_4(n)} = \frac{c_4(n) \times \sigma}{c_4(n)} = \sigma \tag{3.9}$$

So it is a good idea to take as an estimator of σ the average of the $S_i/c_4(n)$:

$$\hat{\sigma} = \frac{1}{k} \sum_{i=1}^{k} \frac{S_i}{c_4(n)} = \frac{\bar{S}}{c_4(n)} \tag{3.10}$$

($\hat{\sigma}$ is unbiased estimator of σ)

This will be, therefore, the formula used for calculating the deviation.

Next, the calculation for one of the deviations needed is performed. It is calculated for the front shaft, explaining the steps followed. These steps are the steps to be followed in order to calculate the other deviations too, which are not written here because it would lengthen the project.

For calculating this deviation only two batches of shafts have been used. The ideal situation would be to have more batches, as this would allow a more accurate calculation of the deviation, but this has not been possible.

Data from these lots are shown in Figure 18:

	1	2
1	25,002	25,003
2	25,003	25,002
3	25,003	25,003
4	25,003	25,003
5	25,002	25,003
6	25,003	25,003
7	25,003	25,003
8	25,004	25,003
9	25,003	25,003
10	25,004	25,003
11	25,003	25,003
12	25,003	25,003
13		25,003
14		25,003
15		25,003

Figure 18: Data from two batches of front shafts produced at the factory

As shown, one of the batches consists of 12 shafts and the other one of 15.

Now the steps just explained theoretically must be applied. It is sought to obtain σ, and as it has seen in (3.10), it is the same as:

$\hat{\sigma} = \frac{1}{k}\sum_{i=1}^{k}\frac{S_i}{c_4(n)} = \frac{\bar{S}}{c_4(n)}$

Therefore $\bar{S}$ and $c_4(n)$ have to be calculated.

The first step to be carried out is the calculation of the average dimension of each batch, or what is the same, $\bar{X}_i$ where $\bar{X}_i = \frac{1}{n}\sum_{j=1}^{k} X_{ij}$

This average $(\bar{X}_i)$ is 25,003 for the batch 1 and 25,00293333 for the batch 2.

The next step is to calculate, for each shaft, S_i applying (3.7).

For these lots it is got:

$S_1 = 0,000603023$

$S_2 = 0,000258199$

Thereby, a result for each batch is calculated. As what is sought is the mean value, it should be added the results and divided the number obtained by the number of batches, which in this case is two.

$\bar{S} = 0,000430611$

Now the value of $c_4(n)$ has to be calculated, which as explained is to be calculated through the formula (3.8).

$c_4(n) = (\frac{2}{n-1})^{1/2} \frac{\Gamma(n/2)}{\Gamma[(n-1)/2]}$

As in the case of $\bar{S}$, this value is calculated for each lot and then calculated the average value as each batch is composed of a different number of elements.

For the batch 1 it's obtained $c_4(12) = 0,977559352$, whereas for the batch 2 it is got $c_4(15) = 0,982316177$. Thus, the average value of $c_4(n) = 0,979937765$.

As the two data needed are already obtained, the deviation for the front shaft can be calculated through the formula (3.10) :

$\hat{\sigma} = \frac{1}{k} \sum_{i=1}^{k} \frac{S_i}{c_4(n)} = \frac{\bar{S}}{c_4(n)} = \frac{0,000430611}{0,979937765} = 0,000439427$

This calculation is very important because this will be the deviation used during the development of the software to simulate the front shaft on the assignment problem, when the nominal dimension of the shaft is calculated.

This is the same process that has to be followed for the calculation of the other deviations, which, like this one, is required at the time of the simulation.

The values obtained are:

- Rear shaft: $\sigma = 0,00042166$
- Shield: $\sigma = 0,000391888$
- Housing: $\sigma = 0,00040336$

After calculating the deviation, the simulation is very simple through a tool like Matlab. This simulation is explained in detail in the section "Strategical approach".

3.2.3 Mathematical modeling of the assembly process as a maximum flow problem

When posing the problem of maximum flow of a network one of the most important things is to establish the scheme that represents our problem, setting the nodes and edges correctly.

On the problem being studied, each diameter represents a subset of the problem and therefore in the diagram. Each subset has as many nodes as parts make up the element's group.

These subsets can be expressed mathematically as:

$$N_i = \{n_{ij}/j = 1, ..., n_i\}, \tag{3.11}$$

where n_i is the number of pieces that make up the element's group i

The set of nodes according to what it has been stated is:

$$Nodes = \{s, t, \bigcup_{i=1}^{n=8} N_i\} \tag{3.12}$$

In the mathematical expression 3.12 n = 8 because it is the number of diameters with which it is worked in order to solve the problem, although the real number of elements is 5. For example, the front diameter and the rear one of the shaft are treated as two different parts but with a very strong connection that is explained later.

Thus, the diagram with which it works is as shown in Figure 19:

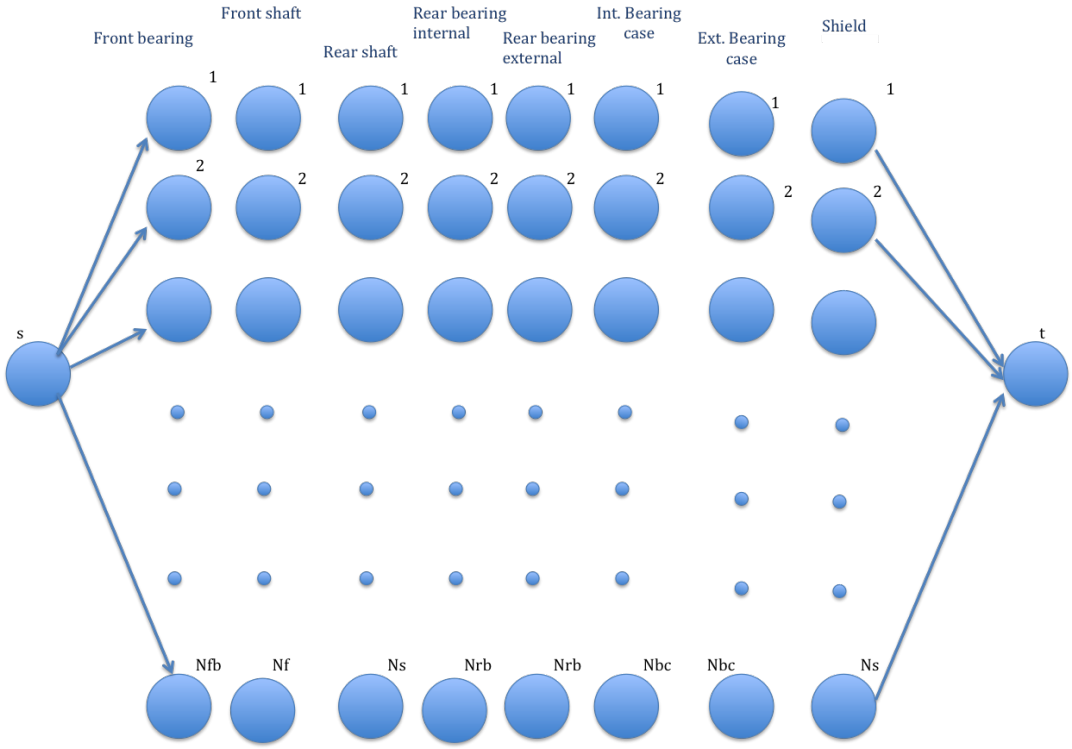


Figure 19: Initial diagram of the maximum flow problem representing all the elements grouped into subsets. Each subset is represented in a column

One of the difficulties of the problem is that the same piece must be related to more than one item at the same time. For example, a bearing is defined simultaneously by its external and internal diameter. Therefore, once the outer diameter is assigned to a housing, the inner diameter is already imposed.

This means that in the diagram there are paths or flows that don't depend on the relationships between the parts, but simply on the parts to be assigned. On the problem being studied the capacity of each arc is unitary, they may never be arriving and departing from a node more than one unit. So this reliance helps to ensure the fulfillment of the law of flow's conservation because it guarantees that from each node comes out, at most, a single flow unit.

On the problem being treated this is reflected as Figure 20 shows:

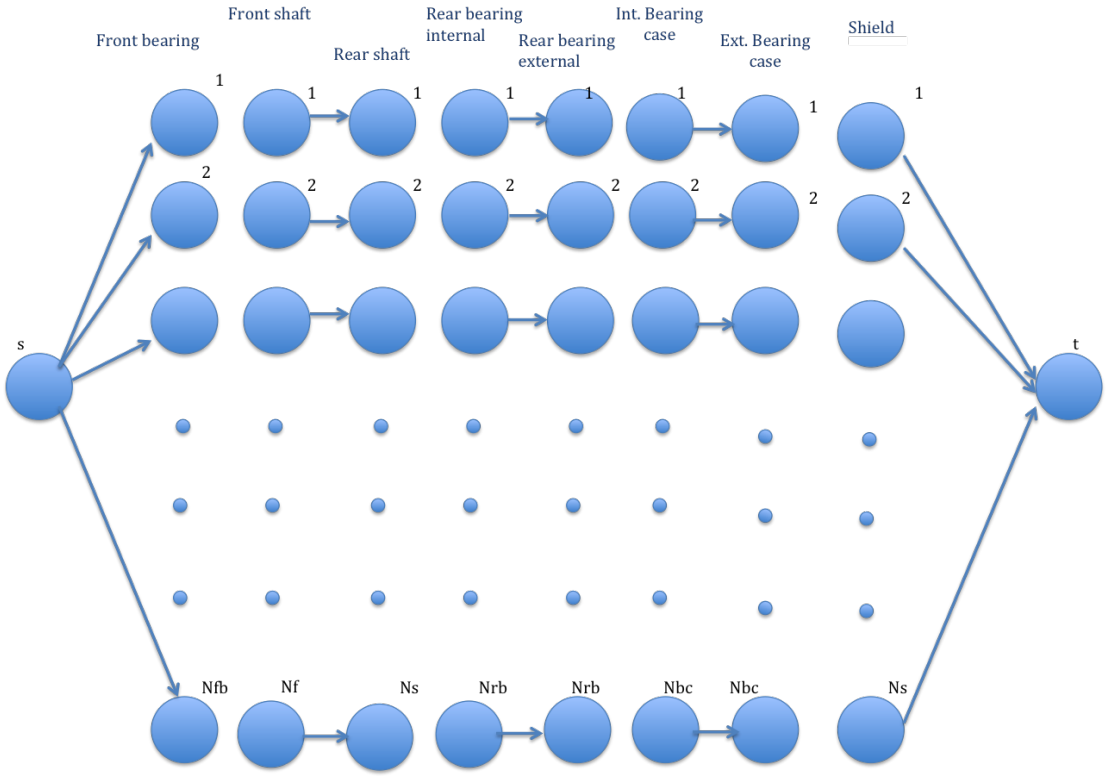


Figure 20: Diagram with the imposed relationships

As far as the rest of relations between parts are concerned, the arcs that represent possible relationships between nodes must be created depending on the relationships between the diameters. Thus, an arc is created if the difference between the diameters of the pieces are within the permissible tolerances (Fig. 14 - Fig. 17). Mathematically:

$$\begin{aligned} &Arcs = \{A_{s1}, \cup_{i=1}^{n=8} A_{i,i+1}, A_{8t}\} \\ &A_{i,i+1} = \{(x,y)/x \in N_i, y \in N_{i+1}\}, \quad \text{where } x \text{ is an acceptable arc} \\ &A_{s1} = \{(s,x)/x \in N_i\} \\ &A_{8t} = \{(x,t)/t \in N_8\} \end{aligned} \tag{3.13}$$

A diagram like Figure 21 is obtained:

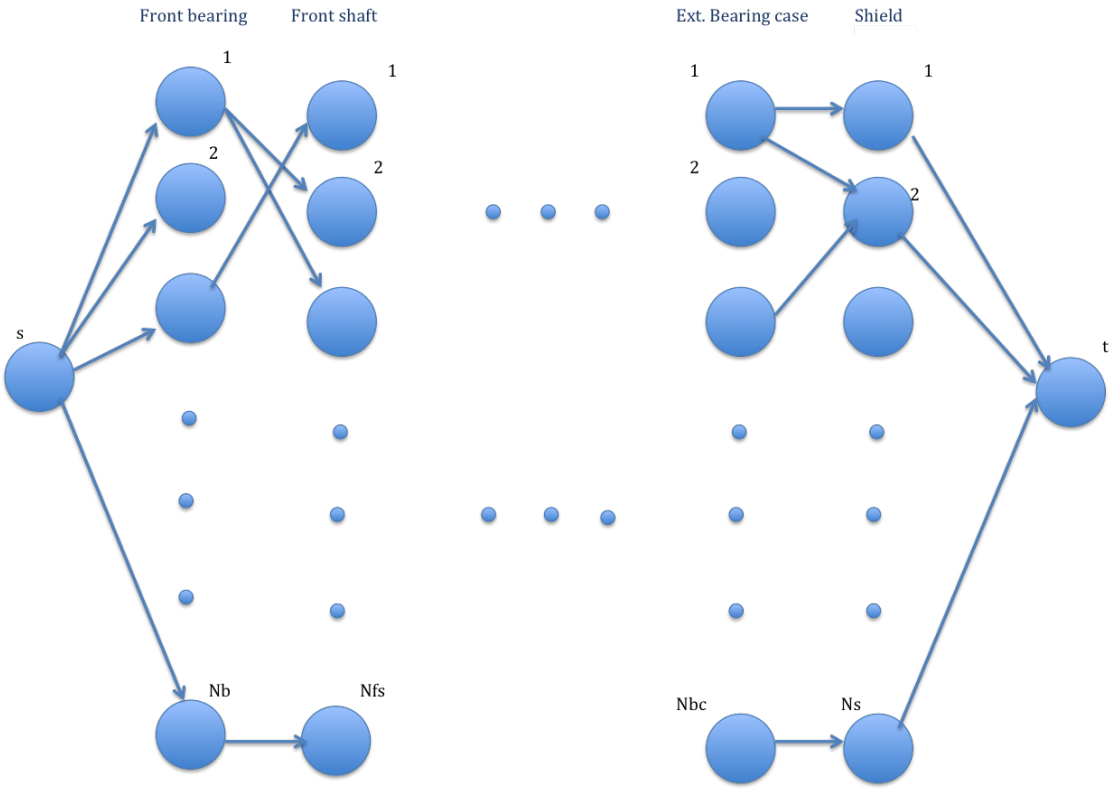


Figure 21: Diagram with all the relationships between pieces

With a diagram of the form of the Figure 21 the problem is solved each time it is performed.

This scheme is the graphical representation of the problem. However, the mathematical representation of it is necessary since it is impossible to solve the problem visually.

In order to represent mathematically the problem, the creation of a NxN matrix is needed, where N is the number of nodes. In that array, the element $a(i, j)$ takes the value:

$$a(i,j) = \begin{cases} 1 & \text{if the arc from node } i \text{ to } j \text{ exists,} \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

As it has already said, an arc is created if the difference between the diameters of the pieces are within the permissible tolerances.

The fact that in this problem the capacity of each arc is unitary means that a matrix composed only by ones and zeros is obtained. By applying the maximum flow' algorithm to this matrix the solution of the problem is obtained.

The mathematical expression of what has just been described can be written as follows:

Maximize v

subject to:

$$\sum_{j\in\Gamma^+(i)} f_{ij} - \sum_{k\in\Gamma^-(i)} f_{ki} = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{if } i \neq s \ \ or \ \ i \neq t \\ -1 & \text{if } i = t \end{cases} \tag{3.15}$$

$$0 \leq f_{ij} \leq 1 \quad \forall (i,j) \in A \tag{3.16}$$

where A represents the set of arcs between nodes among which the flow can be transported; v the quantity of flow on the network from node s to node t and f_{ij} the amount of flow from node i to node j.

3.3 Implementation of the methodology

In the previous section the mathematical modeling required has been explained, or what is the same, the basis necessary to solve the problem in a mathematical way. The next step is therefore the practical development of this modeling. In this project it is performed in Matlab.

In this section the explanation step by step of the mathematical resolution followed is combined with the orders that have been written in Matlab to obtain these results.

3.3.1 Calculation of nominal dimensions

3.3.1.1 Assignment between two parts

In section of mathematical modeling it has been explained that the calculation of the nominal dimension is based on a simulation of many assignments. Before focusing on explaining this simulation, the allocation process is explained.

For the explanation and description of this section just one diameter is used as an example since in all cases the resolution of the problem is the same.

Throughout this section the problem and the resolution is explained based on the relationship between the front shaft and the bearing.

The aim of this assignment is to find the maximum possible number of optimal allocations between the pieces, in this case between the front shaft and the bearing.

As it have been said, this problem is solved using the Hungarian algorithm, which allows the assignment of two groups of parts together.

However, in order to apply this algorithm a cost matrix is needed. In this case, the economic impact of allocating a shaft with one bearing or another is unknown, so this cost matrix can not be calculated.

The matrix that can be obtained from the pieces is a matrix, in which each element is the subtraction of the dimensions of the shafts and the bearings.

What is done therefore is the calculation of this matrix. Shafts are placed in columns and bearings in rows and then its dimensions are subtracted. Since the shafts are larger than the bearings, the subtraction carried out is: shaft diameter - diameter of the bearing, in order to obtain positive numbers.

That's going to be explained with an numerical example, in order to facilitate the method's understanding.

It is supposed to have 5 bearings whose internal diameters are:

25, 24.999, 24.999, 24.998 and 24.997

In the same way, it is supposed to have 5 shafts whose diameters are:

25.004, 25.003, 25.003, 25.003, 25.002

Listing 3.1: Matlab's code to create the matrix substracting the diameters

```
welle=[25.004,  25.003,  25.003,  25.003,  25.002];
lager=[25, 24.999,  24.999,  24.998 and 24.997];

m=numel(welle);
n=numel(lager);
matrix=zeros(m,n);

for i=1:n
  for j=1:m
    matrix(i,j)=welle(j)-lager(i);
  end
end
```

For this example, a matrix with the following values is got:

Listing 3.2: Cost matrix of this example

```
matrix=
0.0020 0.0020 0.0020 0.0030 0.0030
0.0030 0.0030 0.0030 0.0040 0.0040
0.0030 0.0030 0.0030 0.0040 0.0040
0.0040 0.0040 0.0040 0.0050 0.0050
0.0050 0.0050 0.0050 0.0060 0.0060
```

If the Hungarian algorithm is applied to this matrix , it would be a very serious error, since this matrix is not the one that interests us.

What must be done now therefore is a transformation of this matrix just obtained in order to obtain the correct matrix on which to apply the Hungarian algorithm.

This transformation is very simple. It must be first known that, what the Hungarian algorithm seeks is to minimize the objective function. Thus it is given the minimum value , 0, to the admissible combinations. In the case of the front shaft and the bearing, the permissible tolerance is what is among 3 and 5 μm. Accordingly, the values 0,003, 0,004 and 0,005 of the matrix are replaced by 0. The other combinations are not admissible

and thus they are replaced by an infinite value. In this way, these ineligible results will never be a solution.

Continuing with the example developing in Matlab, the code would be:

Listing 3.3: Transformation matrix

```
mat_op=zeros(m,n);
for i=1:m
  for j=1:n
    x=round(matrix(i,j)*1000)/1000;
    if (x==0.005)
     mat_op(i,j)=0;
    end
    if (x==0.004)
      mat_op(i,j)=0;
    end
    if (x==0.003)
      mat_op(i,j)=0;
    end
    if (x>0.005)
      mat_op(i,j)=Inf;
    end
    if (x<0.003)
      mat_op(i,j)=Inf;
    end
  end
end
```

The transformed matrix is in this example:

Listing 3.4: Transformation matrix

```
mat_op=
   Inf   Inf   Inf     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0   Inf   Inf
```

A matrix composed only by 0 and infinite values is obtained. This is the matrix on which the Hungarian algorithm is applied. By applying it to this matrix an assignment that allows to obtain the maximum number of optimal allocations is got.

There is a function in Matlab to implement this algorithm, so it is just written:

$Matching = Hungarian(mat\_op)$

The result obtained is an assignment of shafts and bearings. They are assigned only if the result is acceptable. It won't be assigned those whose relationship is less than 3 or greater than 5 μm.

In this example what is obtained is:

Listing 3.5: Solution matrix after applying the Hungarian algorithm

```
Matching=

     0     0     0     1     0
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     0     0
```

The elements on which rolls a 1 are those in which an acceptable allocation is obtained. This means that the bearing i and the shaft j must be assigned in order to obtain the largest possible number of optimal combinations.

To show the results in Matlab it should be written the following.

Listing 3.6: Matlab code to show the solutions

```
welle_vec=zeros(num_data,1);
lager_vec=zeros(num_data,1);
cont_welle=1;
cont_lager=1;
opt=0;
for i=1:m
  for j=1:n
    if (mat_op(i,j)==0) &&(Matching(i,j)==1)
      opt=opt+1;
       disp('The following shaft and bearing fit together with
           optimal play:');
       Welle=j
       Lager=i
       welle_vec(cont_welle)=welle(j);
       lager_vec(cont_lager)=lager(i);
```

```
        cont_welle=cont_welle+1;
        cont_lager=cont_lager+1;
    end
  end
end
```

This way it is shown:

Listing 3.7: Solutions of the problem

```
The following shaft and bearing fit together with optimal play:
Shaft =

     4
Bearing =

     1
The following shaft and bearing fit together with optimal play:
Shaft =

     1

Bearing =

     2

The following shaft and bearing fit together with optimal play:
Shaft =

     2

Bearing =

     3

The following shaft and bearing fit together with optimal play:

Shaft =

     3
Bearing =

     4
```

It means that the following assignments must be carried out:

Assignment 1: Shaft 4 (25.003) - Bearing 1 (25)

Assignment 2: Shaft 1 (25.002) - Bearing 2 (24.999)

Assignment 3: Shaft 2 (25.002) - Bearing 3 (24.998)

Assignment 4: Shaft 3 (25.002) - Bearing 4 (24.998)

Not assigned: Shaft 5 (25.003) ; Bearing 5 (24.997)

It can be stated that the solution obtained is the best possible solution for that dataset.

3.3.1.2 Calculation of nominal dimensions

The calculation of the nominal values is performed based on the above problem.

The problem of the calculation of nominal dimensions is solved through a simulation, which allows to know which results could be obtained in the case that the different diameters are produced but without produce them.

Specifically what is done is the simulation 500 times of the production for each possible diameter. In order to perform this simulation just the dimension and the deviation (calculated in "mathematical modeling") are needed. In each of these 500 simulations the assignment by the Hungarian algorithm is also simulated and the number of possible combinations is thus obtained.

To compare the results obtained with each diameter, a numeric indicator is needed, which in this case is the mean value. For each simulation a number of assignments is obtained. The average of these allocations is calculated for each diameter simulated and then these values are compared with each other. The diameter, for which a largest average value is obtained, is the one that must be produced.

As in the previous case, the procedure is explained with a numerical example to facilitate its comprehension.

The same data as in the example above are used.

Diameters of the bearings: 25, 24.999, 24.999, 24.998 and 24.997

For this problem, four possible shaft diameters have to be simulated: 25.001, 25.002, 25.003, 25.004.

This is so because for the shaft to be admissible, it must be between 25,001 and 25,004 and because the precision with which the company can measure the shafts is μm. If a greater precision was possible, more diameters would be simulated or the data should be treated as continuous distributions.

To simulate the different shafts it's programmed in Matlab:

Listing 3.8: Matlab's code

```
lager=[25 24.999  24.999  24.998 24.997];
dec=0;
num_sim=500;
c=4;
mat_ana=zeros(num_sim,c);

acum_fin=0;
sol_fin=0;
eje=0;

for c=1:4
    acum=0;
    pos=0;
    sol=0;
    for k=1:num_sim
       N=numel(lager);
% Front shaft can be between 25.001 and 25.004
       mean_d3=25.001+dec
       stdd_d3=0.000439427;
       d3=randn(N,1).*stdd_d3+mean_d3;

d3_r=round(d3*1000)/1000;
welle=[d3_r];

m=numel(welle);
n=numel(lager);
matrix=zeros(m,n);

for i=1:m
  for j=1:n
    matrix(i,j)=welle(j)-handles.lager(i);
```

```
  end
end

mat_op=zeros(m,n);
```

Now it is written as in the case of the assignment problem because as it has been said for this problem the same simulation is carried out many times:

Listing 3.9: Matlab's code to create the transformation matrix and apply the Hungarian algorithm

```
for i=1:m
  for j=1:n
    x=round(matrix(i,j)*1000)/1000;
    if (x==0.005)
     mat_op(i,j)=0;
    end
    if (x==0.004)
      mat_op(i,j)=0;
    end
    if (x==0.003)
      mat_op(i,j)=0;
    end
    if (x>0.005)
      mat_op(i,j)=Inf;
    end
    if (x<0.003)
      mat_op(i,j)=Inf;
    end
  end
end

Matching = Hungarian(mat_op);

opt=0;
for i=1:m
  for j=1:n
    if (mat_op(i,j)==0) &&(Matching(i,j)==1)
      opt=opt+1;
    end
  end
end

kein=N-opt
```

What is done now is a creation of a matrix of 4 columns, one for each simulated shafts and 500 rows, one for each simulation. Each element of this array is the number of assignments obtained for the corresponding simulation.

$sum = opt;$
$mat\_ana(k, c) = sum;$

Once the simulation of a shaft is carried out as many times as necessary, another should be simulated. To do that it is written:

$dec = dec + 0.001;$

To analyze the results in a visual way, a box plot is shown:

Listing 3.10: Matlab's code to show the resultsi n a boxplot

```
[Pr, ANOVATABr, STATSr]=anova1(mat_ana);
[cr,mr,hr,nmar]=multcompare(STATSr);
boxplot(mat_ana,{'25.001','25.002','25.003','25.004'})
```

The box-plot obtained for this example is shown in Figure 22:

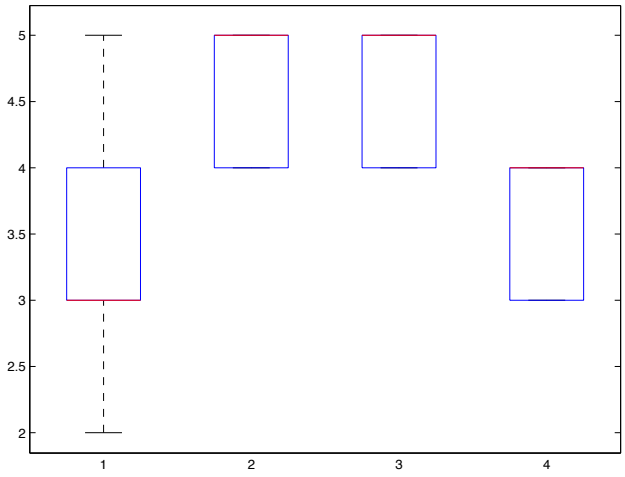


Figure 22: Obtained box-plot for this example

As explained, in order to compare the results obtained for each simulated diameters, the mean value is calculated. To do that and compare the values

obtained, so that the largest is got, in Matlab:

Listing 3.11: Matlab's code to show the results

```
media_fin=0;
for i=1:c
    suma_fila=0;
    cont=0;
    for j=1:k
        suma_fila(c)=mat_ana(j,i)+cont;
        cont=suma_fila(c);
    end
    suma_fin=suma_fila(c)/k
    if suma_fin>media_fin
        media_fin=suma_fin
        eje=i;
    end
end

sol_eje=0;
disp('La mejor combinacion se obtiene para:');
if eje==1
   disp('eje de diametro 25.001');
   sol_eje=25.001;
end

if eje==2
   disp('eje de diametro 25.002');
   sol_eje=25.002;
end

if eje==3
   disp('eje de diametro 25.003');
   sol_eje=25.003;
end

if eje==4
   disp('eje de diametro 25.004');
   sol_eje=25.004;
end
```

The following is displayed in Matlab:

"The best result is obtained for: Diameter 25.002"

Although in the box-plot seems that the same average is obtained for a diameter of 25,002 mm. and for a diameter of 25,003 mm., it must be

stressed that what is got is:

```
Mean value for diameter 25.002mm:   4.7200

Mean value for diameter 25.003mm:   4.5380
```

According to the result, a shaft of 25,002 mm diameter should be produced so that the greatest number of assignments are obtained, depending on the bearings conforming this lot.

3.3.1.3 Summary

In order to summarize what has been explained, Figure 24 is attached. In this Figure a scheme is shown with the main and chronological steps that must be followed according to the procedure developed. If these steps are followed, the right solution of the problem is achieved.

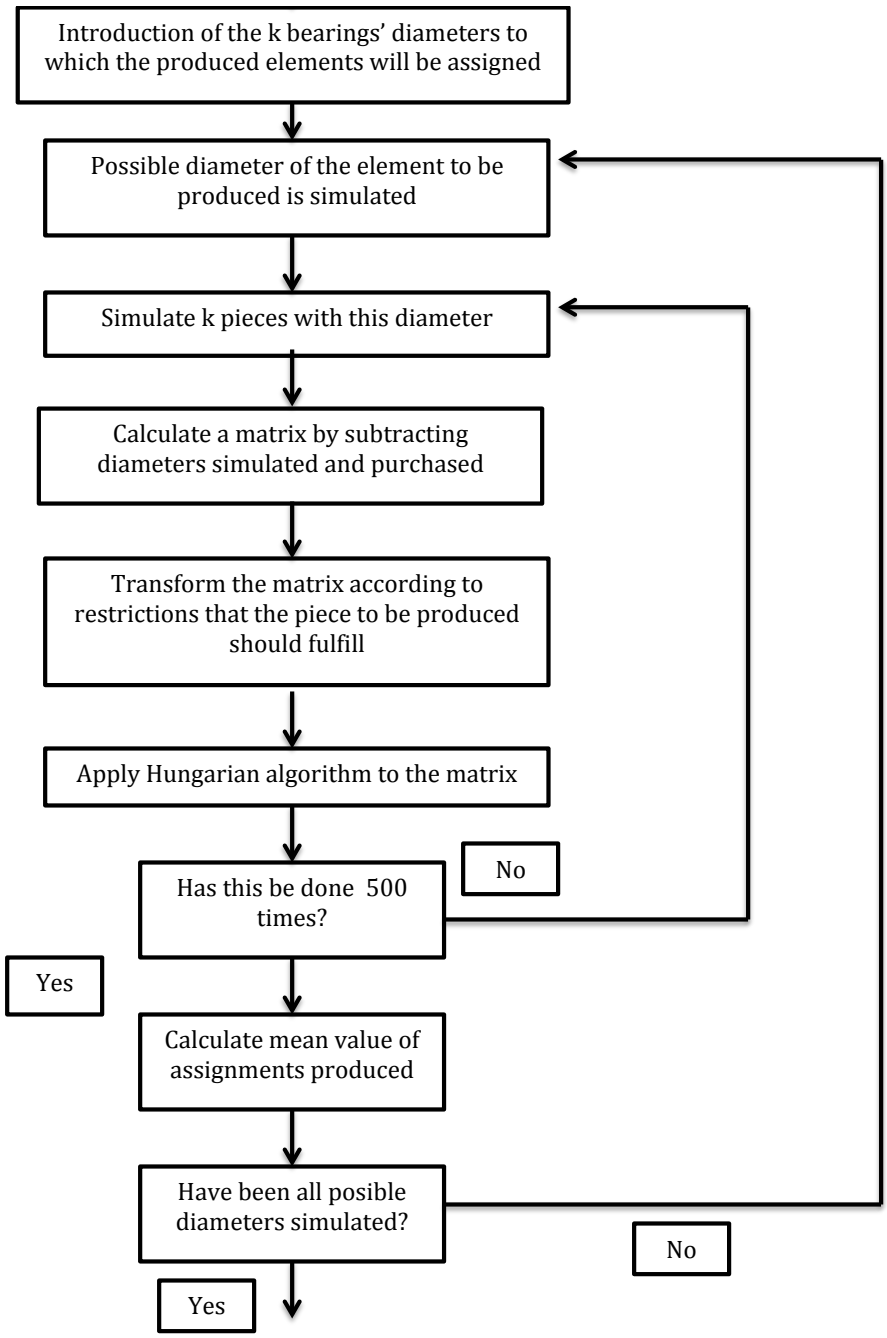


Figure 23: Scheme with the steps to be followed

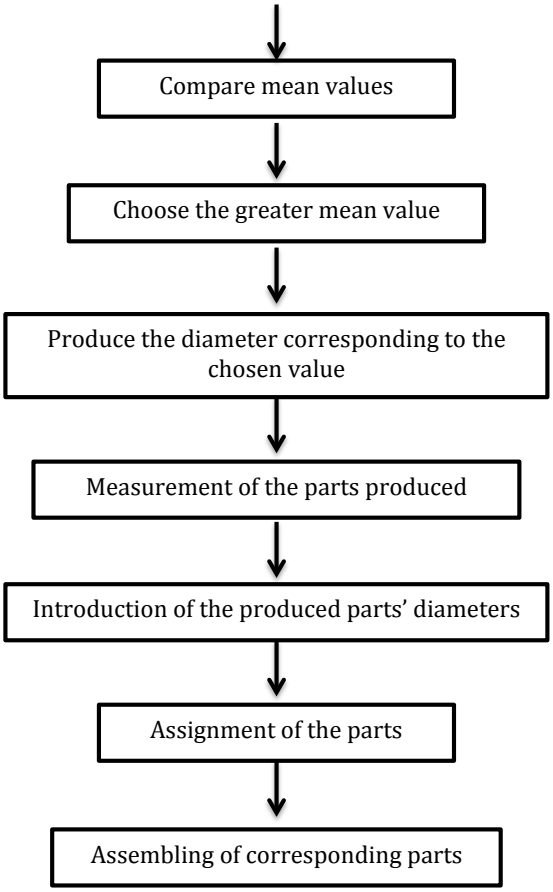


Figure 24: Scheme with the steps to be followed(2)

3.3.2 Assignment problem

The second problem that the company faces is the assembly of all the pieces simultaneously, once all have been produced and are available.

The first step is the introduction of the data for each group of items, paying attention to the fact that those diameters belonging to the same part shall take the same position in the respective groups. That is, if the shafts to be assigned have the following dimensions: 25.004-22.003; 25.004-22.004; 25.003-22.003, these data should be entered as follows:

```
Front shaft = [25,004 25,004 25,003];
Rear shaft= [22.003 22.004 22.003];
```

ensuring that the first data of the two groups belong to the same shaft, as in the case of the second and so on. That should be applied to the rear bearing and the bearing case too.

Let's suppose that the data entered are the following:

Listing 3.12: Data used foe the example

```
Front bearing: rd=[25 24.999 24.998 25];
Front shafts: ed=[25.004 25.004 25.003];
Rear shaft: et=[22.003 22.004 22.003];
Internal rear bearing: rt_i=[22 21.999 22 21.999];
External rear bearing: rt_e=[42 41.999 41.998 42];
Internal bearing case: carc_i=[42.006 42.008 42.008 42.007];
External bearing case: carc_e=[60 60 60.001 60];
Shield: cub=[60.001 60 60.001 60.002 60];
```

Understandably, the number of front and rear shafts should be the same, as in the case of internal and external rear bearings or the internal and external bearing cases.

Values for the source node and the sink node are also introduced.

```
Source node: s=1;
Sink node: t=1;
```

The number of nodes in this example are:

$nodos = n\_s + n\_rd + n\_ed + n\_et + n\_rt_i + n\_rt_e + n\_carc\_i + n\_carc\_e + n\_cub + n\_t$;

$nodos = 33$;

As explained in the theoretical basis a matrix of dimensions Nodes x Nodes is needed for resolution.

This array consists of x_{ij} elements whose values are:

$$x_{ij} = \begin{cases} 1 & \text{if the arc between node } i \text{ and node } j \text{ is feasible} \\ 0 & \text{if the arc between node } i \text{ and node } j \text{ is not feasible} \end{cases} \quad (3.17)$$

To know whether these arcs are feasible or not, the diameters of those parts related to each other are subtracted and it has to be considered whether this subtraction is within acceptable tolerances.

In Matlab this is done as follows:

Since there is no relationship between the source node and the front bearings, an arc is created from the source node to each bearing.

Listing 3.13: Relathionship source node-front bearing

```
%relathionship source node-front bearing
i=1:n_s
    for j=1:(n_rd+1)
        matrix(i,j)=1;
    end
end
```

In the case of the front bearing and the front shaft, the substraction of diameters should be between 0.003 and 0.005 mm. as already explained in the section of product explanation. If the substraction of the front shaft (j) and the front bearing (i) is within this range, "1" is placed in the element x_{ij} otherwise a "0".

Listing 3.14: Relationship front bearing - front shaft

```
%relationship front bearing - front shaft
for i=1:n_rd
    for j=1:n_ed
        resta=ed(j)-rd(i);
        x=round(resta*1000)/1000;
        if (x>=0.003)||(x<=0.005)
            matrix(i+1,(n_rd+1)+j)=1;
        end
        if (x<0.003)||(x>0.005)
            matrix(i+1,(n_rd+1)+j)=0;
        end
    end
end
```

This should also be applied to relationships between rear shaft - internal rear bearing; external rear bearing-internal bearing case and internal bearing case-internal shield.

Once a shaft has been assigned, the diameter of the rear shaft is already imposed. For this to be true is written as follows:

Listing 3.15: Relationship front shaft -rear shaft

```
%relationship front shaft -rear shaft
for i=1:n_ed
  matrix(i+(n_s+n_rd),(n_s+n_rd+n_ed)+i)=1;
end
```

This should also be applied in the case of rear bearing (external and internal) and the bearing case (internal and external).

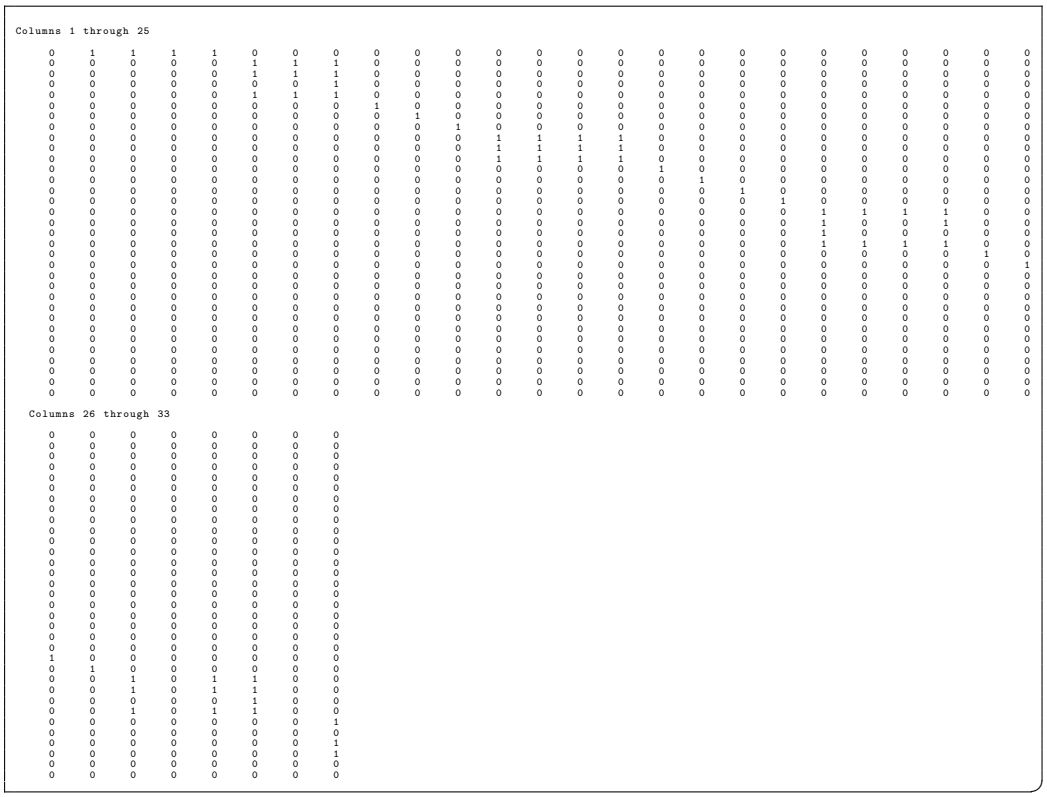
For the relationship between the housing and the sink node:

Listing 3.16: Relationship shield-sink node

```
%relationship shield-sink node
cont=1;
for j=1:n_cub
    suma=0;
    for i=1:n_carc_e
       suma=suma+matrix(n_s+n_rd+n_ed+n_et+n_rt_i+n_rt_e+
          n_carc_i+i,j+n_s+n_rd+n_ed+n_et+n_rt_i+n_rt_e+
          n_carc_i+n_carc_e)
    end
    if suma~=0
       matrix(j+(n_s+n_rd+n_ed+n_et+n_rt_i+n_rt_e+n_carc_i+
          n_carc_e),(n_s+n_rd+n_ed+n_et+n_rt_i+n_rt_e+n_carc_i+
          n_carc_e+n_cub+n_t))=1;
    end
end
```

Thus a "1" is written in the last column (j=nodes) (corresponding to sink node) if at the node (i) (corresponding to the shields) arrives at least one arc. If that's not the case, "0" is written.

The obtained matrix (33x33) is in this example:



When resolving in Matlab a call of an existing function is necessary. This function is "graphmaxflow" and that's carried out as follows:

$$[MaxFlow, FlowMatrix, Cut] = graphmaxflow(G, SNode, TNode)$$

"Graphmaxflow" calculates the maximum flow of directed graph G from node SNode to node TNode. Input G is an N-by-N sparse matrix that represents a directed graph. Nonzero entries in matrix G represent the capacities of the edges. Output MaxFlow is the maximum flow, and FlowMatrix is a sparse matrix with all the flow values for every edge. FlowMatrix(X,Y) is the flow from node X to node Y. Output Cut is a logical row vector indicating the nodes connected to SNode after calculating the minimum cut between SNode and TNode. If several solutions to the minimum cut problem exist, then Cut is a matrix.

So, before applying the function, the matrix obtained is defined as a sparse matrix.

Listing 3.17: Calculation of maximum flow

```
cm = sparse(matrix);
[M,F,K] = graphmaxflow(cm,1,nodos);
```

The result obtained after applying the function is:

Listing 3.18: Solution obtained

```
M =

     3
F =

   (1,2)         1
   (1,4)         1
   (1,5)         1
   (2,6)         1
   (5,7)         1
   (4,8)         1
   (6,9)         1
   (7,10)        1
   (8,11)        1
   (9,12)        1
  (11,13)        1
  (10,15)        1
  (12,16)        1
  (13,17)        1
  (15,19)        1
  (16,20)        1
  (19,21)        1
  (17,23)        1
  (20,24)        1
  (21,25)        1
  (23,27)        1
  (27,28)        1
  (25,30)        1
  (24,31)        1
  (28,33)        1
  (30,33)        1
  (31,33)        1
```

It is known that the maximum spindles to be obtained with the parts are 3 (which it is also the maximum number possible, as it is the number of shafts) and nodes used for this purpose are obtained too. However, it's not possible to know the paths, the relationships between parts necessary

for obtaining those spindles.

To obtain these relationships the following has to be written:

Listing 3.19: Code to obtain the paths

```
cont=1;
for i=1:n_rd
  from4= graphtraverse(F,i+1);
  to1= graphtraverse(F',nodos);
  h= intersect(from4,to1)
  F2 = F(h,h);
  TF = isempty(h);
  if TF==0
    mat_sol(cont,:)=h(1,:);
    if cont<M
        cont=cont+1;
    end
  end
end
```

h provides the different solution paths of the problem, which in this case are:

Listing 3.20: Obtained paths for this example

```
h =
     2     6     9    12    16    20    24    31    33
h =
     4     8    11    13    17    23    27    28    33
h =
     5     7    10    15    19    21    25    30    33
```

These paths are accumulated in a matrix as follows:

Listing 3.21: Matrix with the paths

```
mat_sol =

     2     6     9    12    16    20    24    31    33
     4     8    11    13    17    23    27    28    33
     5     7    10    15    19    21    25    30    33
```

However, this is not useful because it's just known the number of the

node but not the part to which that node refers. For this to be solved a vector that accumulates the values of the parts in the order of appearance is created. Thus, the node number and dimension correspond to each other.

$$valores = [s \;\; rd \;\; ed \;\; et \;\; rt\_i \;\; rt\_e \;\; carc\_i \;\; carc\_e \;\; cub \;\; t];$$

Listing 3.22: Transformation matrix to get the values of the solution paths

```
for i=1:M
   for j=1:9
      indice=mat_sol(i,j);
      mat_sol(i,j)=valores(indice);
   end
end
```

Finally a matrix is obtained in which the dimensions located in each row are those assigned to produce a final spindle.

```
mat_sol =

  Columns 1 through 8

   25.0000   25.0040   22.0030   22.0000   42.0000   42.0060   60.0000
       60.0020
   24.9980   25.0030   22.0030   21.9990   41.9990   42.0070   60.0000
       60.0010
   25.0000   25.0040   22.0040   21.9990   42.0000   42.0080   60.0000
       60.0010
```

So, in this case, to obtain the maximum number of spindle correctly produced parts should be assembled as follows:

25.0000 → 25.0040 → 22.0030 → 22.0000 → 42.0000 → 42.0060 → 60.0000 → 60.0020

24.9980 → 25.0030 → 22.0030 → 21.9990 → 41.9990 → 42.0070 → 60.0000 → 60.0010

25.0000 → 25.0040 → 22.0040 → 21.9990 → 42.0000 → 42.0080 → 60.0000 → 60.0010

Furthermore, the use of graphmaxflow allows a visualization of the diagram that is created for the resolution of the problem through the order "biograph". Function BGonj=biograph(CMatrix) creates a biograph object, BGobj, using a connection matrix, CMatrix. All nondiagonal and positive

entries in the connection matrix, CMatrix, indicate connected nodes, rows represent the source nodes, and columns represent the sink nodes.

Listing 3.23: Code to get the initial diagram

```
bg = biograph(cm,'LayoutType','hierarchical','ShowTextInNodes',
    'Label');
view(bg)
```

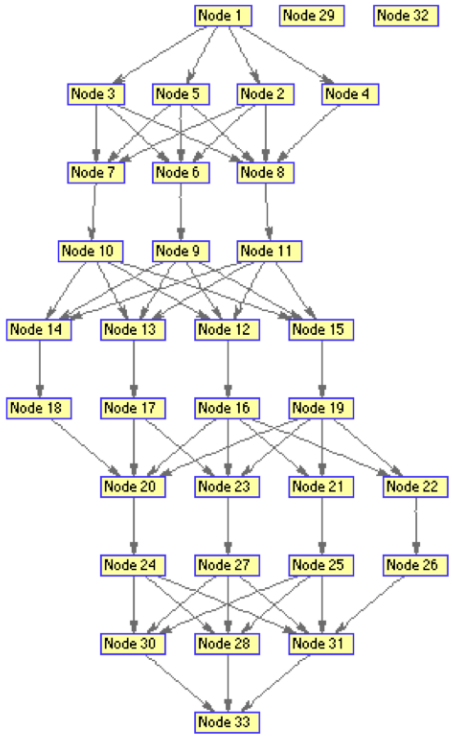


Figure 25: Initial diagram of the problem to be solved

For the solution's diagram it is written:

Listing 3.24: Code to get the initial diagram

```
solucion=biograph(F,'LayoutType','hierarchical','
    ShowTextInNodes','Label')
view(solucion)
```

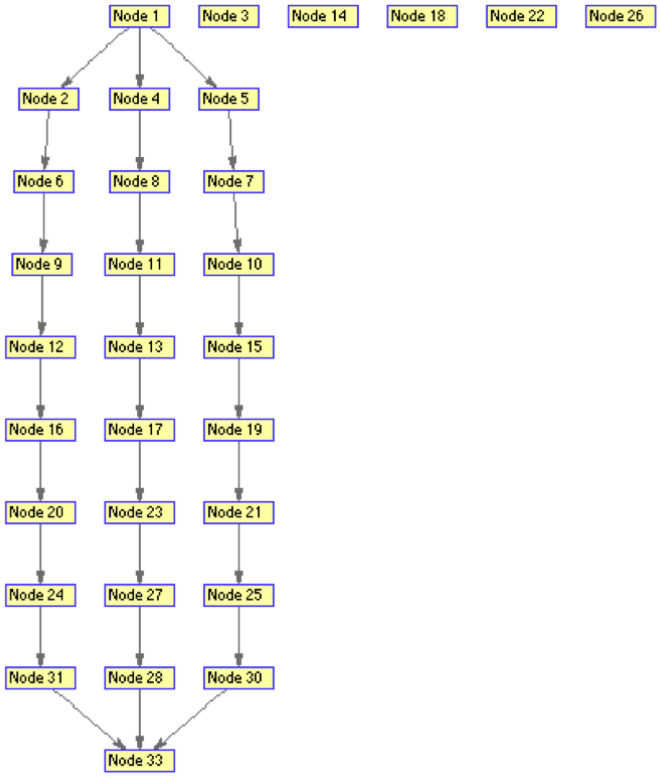


Figure 26: IDiagram of the solution

3.4 Software's description

Once that the mathematical procedure has been achieved and the resolution of both problems that were faced at the beginning of the project is now possible, it is necessary to develop a computer's software that allows in a simple and fast way the implementation of that procedure in the company.

In order to get that, a simple Graphical User Interface (GUI) is built. The user is allowed to introduce the dimensions of the parts with which he/she works for the simulation by menus and dialog windows displayed by the GUI.

It is a simple and intuitive interface that requires no mathematical knowledge to its use.

In this section its operation and its appearance are explained and shown.

First thing that comes out when the execution of the program is a menu with two options, which looks like Figure 27.

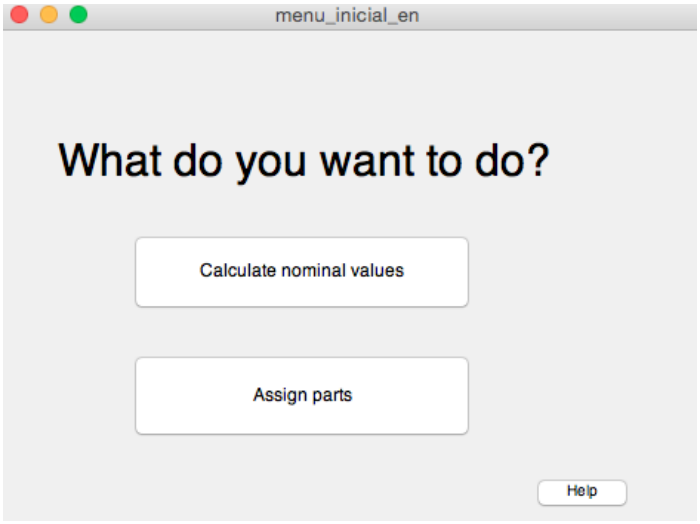


Figure 27: Software's initial menu

The option that interests at that time should be clicked: calculate nominal values or assign parts.

3.4.1 Calculate nominal values

If the option "calculate nominal values" is chosen, the window shown in Figure 28 opens.

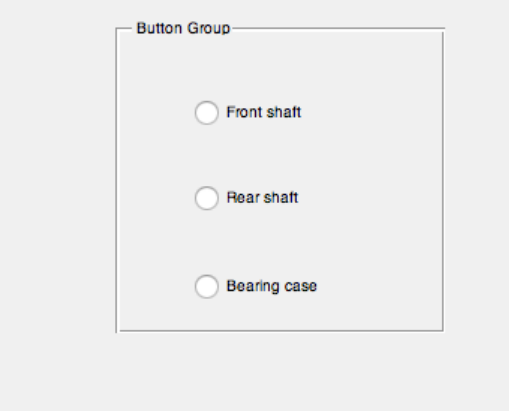


Figure 28: Displayed menu after selecting "Calculate nominal values"

In the options menu that appears, the four diameters that can be calculated can be seen: front shaft, rear shaft, housing and shield.

The following steps are the same regardless of which option it's chosen, so all is explained only for one case.

Following the developing example along the explanation, the front shaft is chosen.

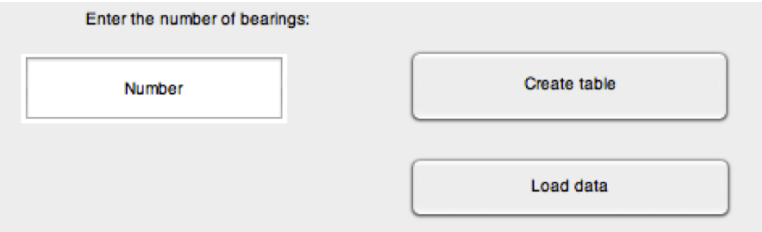


Figure 29: Options to entered the data

In the next window that opens, which Figure 29 shows, two options can be chosen: enter the number of bearings, if some bearings are being used for the first time, or load some data, if working with some bearings that were not assigned previously

In the first case, the corresponding number is entered and then the button "create table" is pressed.

In the second case, the button "load data" should be pressed and then the file in which the bearings are collected selected.

Either way, a table of Figure 30 appears. If a data file has been selected, bearings will be written in the table. If not, their dimensions should be inserted. In case they have to be inserted:

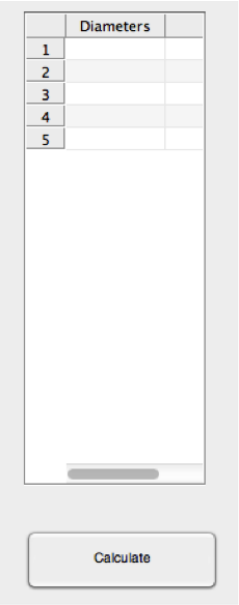


Figure 30: Displayed table to insert the data

In the table, the diameters of the bearings are introduced. These are:

25, 24.999, 24.999, 24.998 and 24.997

When the "calculate" button is clicked the following window that Figure 31 shows is displayed:

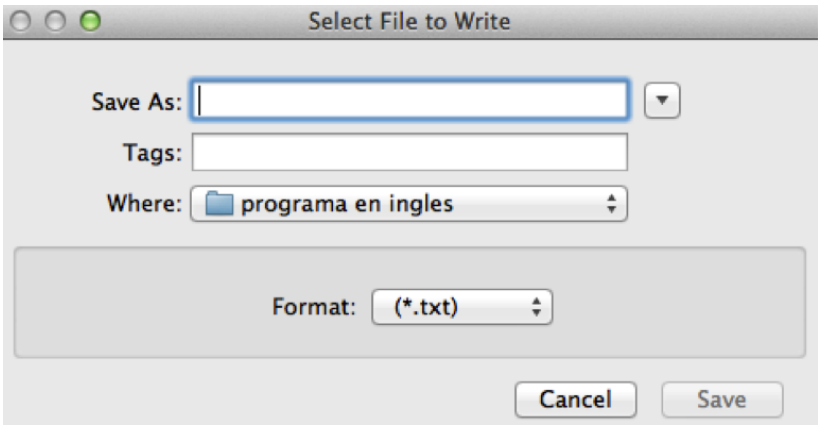


Figure 31: Displayed window to save the data file

This way, the entered bearing's data can be saved, choosing the name of the file, its format and its location, thus facilitating data storage for the company.

The final window that is displayed looks like Figure 32.

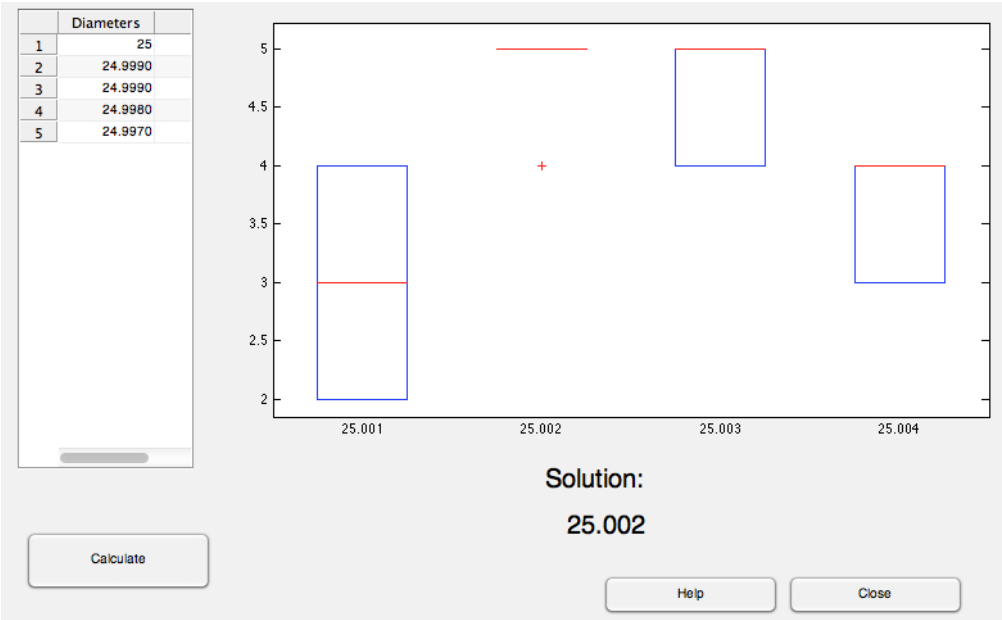


Figure 32: Window where the final results are shown

One box-plot, like in Figure 32, is shown, where a quick visual analysis of the results can be performed, which can be very useful when analyzing them, because it goes without any mathematical knowledge or long time to carry it out.

As already explained, the diameter whose average mean, represented in box-plot by the red line, is greater must be chosen.

Also a numerical solution is attached to facilitate the obtaining of the result and so that there are no doubts.

3.4.2 Assignment

If the option "Assign parts" is chosen, the window shown in Figure 33 opens.

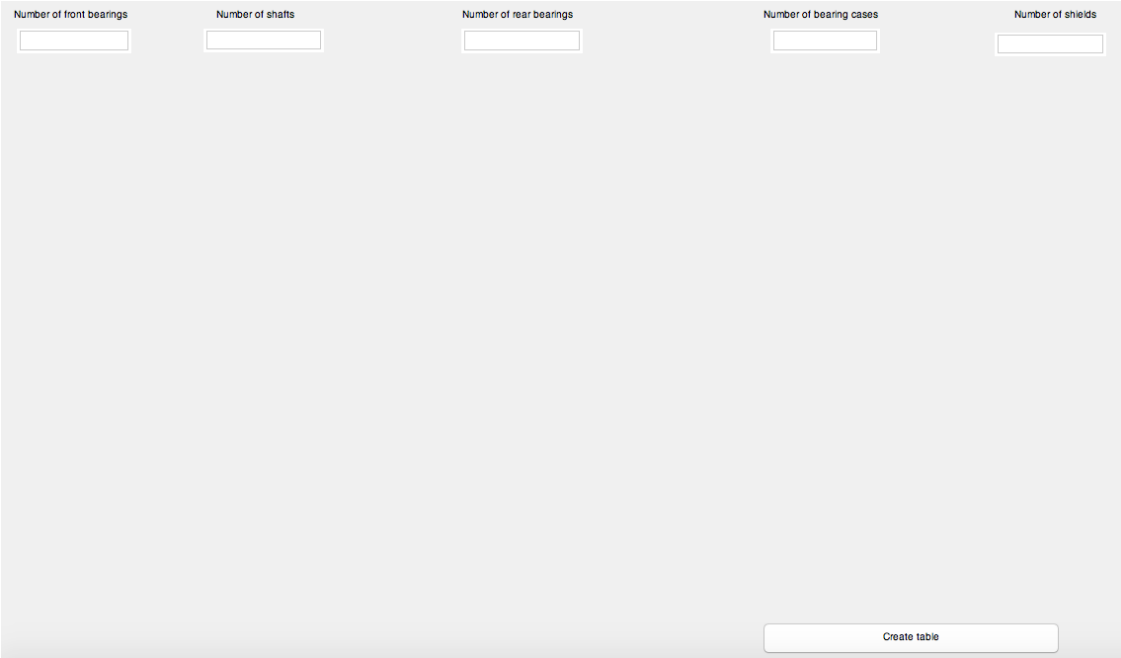


Figure 33: Displayed window where the number of available pieces should be inserted

For this example it's suppose the following pieces are available:

Listing 3.25: Data used foe the example

```
Front bearing: rd=[25 24.999 24.998 25];
Front shafts: ed=[25.004 25.004 25.003];
Rear shaft: et=[22.003 22.004 22.003];
Internal rear bearing: rt_i=[22 21.999 22 21.999];
External rear bearing: rt_e=[42 41.999 41.998 42];
Internal bearing case: carc_i=[42.006 42.008 42.008 42.007];
External bearing case: carc_e=[60 60 60.001 60];
Shield: cub=[60.001 60 60.001 60.002 60];
```

That's means that the numbers that should be entered are:

```
Front bearing: rd=4;
Shafts: ed=3;
Rear bearing: rt=4;
Bearing case: carc_i=4;
Shield: cub=5;
```

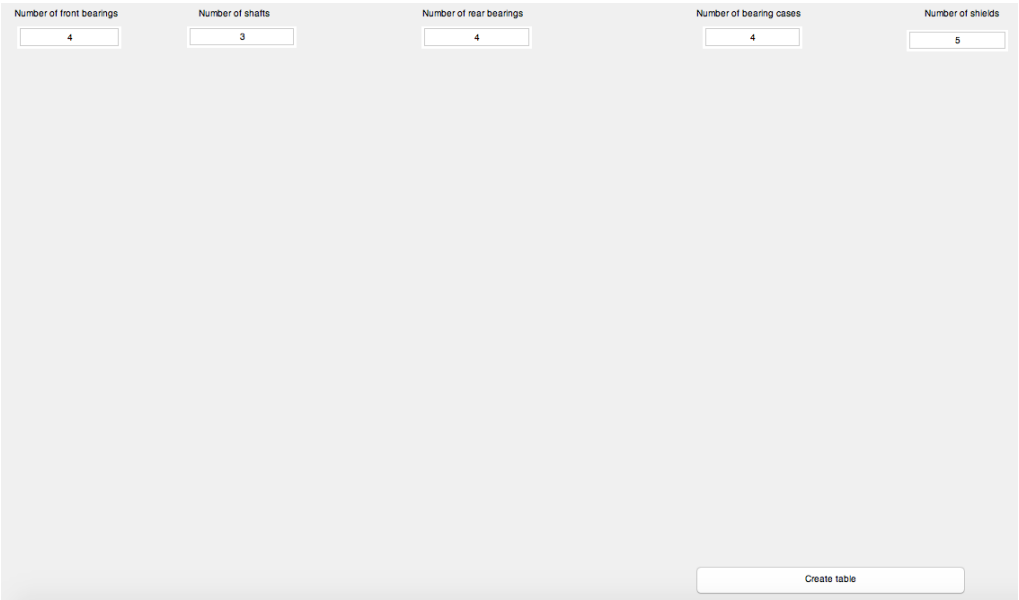


Figure 34: Numbers inserted

Once the numbers of the items are inserted as in Figure 35, the button "Create table" is pressed and the window as in Figure is shown.

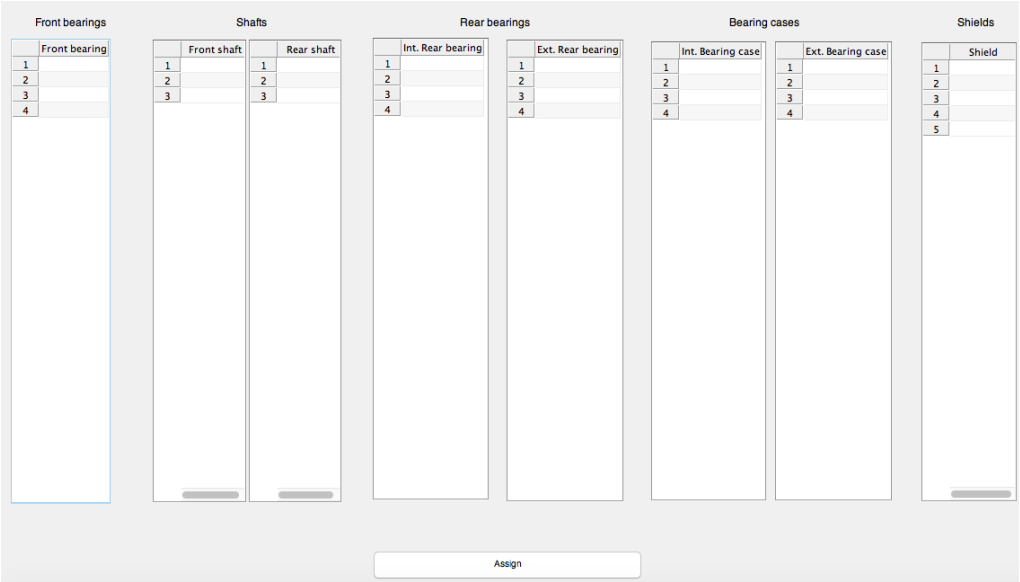


Figure 35: Tables to insert the dimensions of the pieces to be assigned

In these tables the dimensions of the available pieces are inserted in order to be assigned.

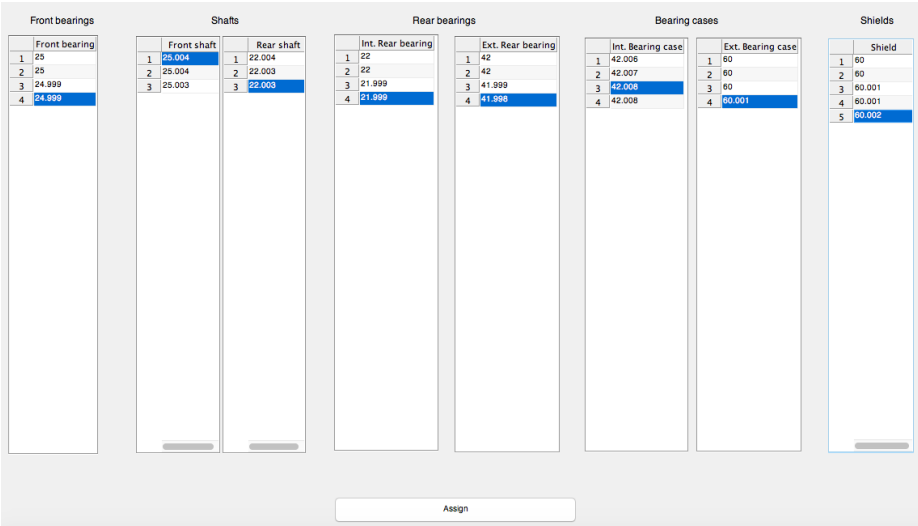


Figure 36: Tables to insert the dimensions of the pieces to be assigned

After inserting the data, button "Assing" is pressed and the a window like Figure 37 is displayed:

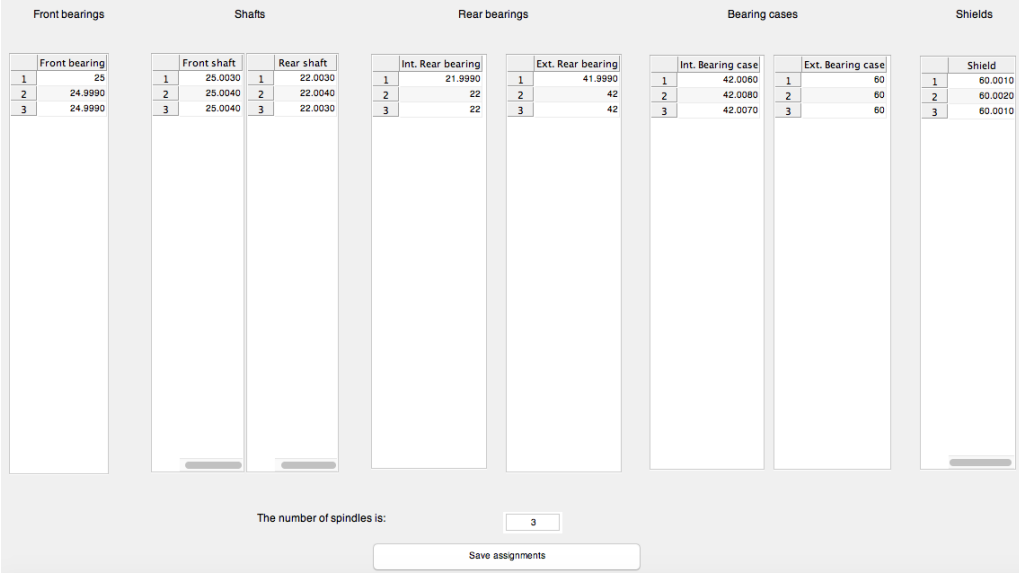


Figure 37: Obtained assignments

Pieces located in the same row are the ones that should be assigned together.

At the bottom of the window the number of obtained combinations is shown as well as a button that allows to save theses combinations in different formats as it's shown in the Figure

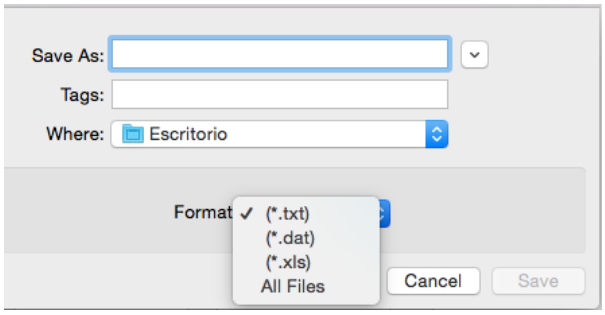


Figure 38: Displayed window to save the final combinations

Chapter 4

Results

In this section some examples of simulation are carried out and next their results are analyzed in detail.

Since the results obtained in each execution depend on the data entered, it is not possible to conduct a general analysis of the results.

What is done is a comparison of what is got in three different situations. In the first one, no nominal dimension is calculated; in the second one, just one nominal dimension is calculated and in the last one, the three possible diameters have been calculated and the parts have been produced according to these diameters.

4.1 Results for some examples

In these example, to be as close to reality as possible, real data from batches, with which the company has worked, are used.

Listing 4.1: Real data used

```
rd =  [ 25.0000  25.0000 24.9990 25.000  24.9990 24.9980
    25.0000 25.0000 24.9990 24.9980 24.9970 25.0000 24.9990
    25.0000 24.9980 24.9970 25.0000 25.0000 24.9990 24.9970
    25.0000 25.0000 24.9990 24.9980 25.0000 24.9990 24.9980
    24.9990  25.0000];

ed =[ 25.0030 25.0040 25.0030 25.0030 25.0020 25.0040 25.0030
    25.0030 25.0030 25.0040 25.0020 25.0040 25.0030 25.0040
    25.0040 25.0030 25.0030 25.0040 25.0020 25.0030 25.0030
```

```
    25.0040 25.0020 25.0040 25.0030 25.0030  25.0020  25.0040
    25.0030];

et =[22.0040 22.0040 22.0040 22.0030 22.0020 22.0030 22.0040
    22.0040 22.0020 22.0040 22.0020 22.0040 22.0040 22.0030
    22.0040 22.0040 22.0030 22.0040 22.0040 22.0020 22.0030
    22.0040 22.0040 22.0030 22.0020 22.0040 22.0020 22.0030
    22.0040];

rt_e =[ 42.0000 41.9990 41.9980 42.0000 42.0000 41.9990 42.0000
     41.9990 41.9970 42.0000 42.0000 41.9990 42.0000 41.9970
    42.0000 41.9970 41.9990 42.0000 41.9980 42.0000 41.9980
    41.9990 42.0000 41.9980 42.0000 41.9980 41.9990  42.0000
    41.9980];

rt_i =[ 21.9990 22.0000 22.0000 21.9980 21.9990 21.9990 21.9980
     21.9990 22.0000 22.0000 22.0000 21.9990 22.0000 21.9980
    22.0000 21.9990 21.9990 21.9990 22.0000 21.9980 22.0000
    22.0000 21.9990 21.9990 21.9990 21.9990 21.9990 21.9990
    21.9990];

carc_i =[42.0070 42.0080 42.0070 42.0070 42.0070 42.0070
    42.0080 42.0060 42.0070 42.0070 42.0070 42.0060 42.0060
    42.0080 42.0070 42.0070 42.0060 42.0070 42.0070 42.0060
    42.0080 42.0070 42.0070 42.0060 42.0070 42.0080 42.0080
    42.0070 42.0080];

carc_e =[60.0000 60.0010 60.0000 60.0010 60.0000 60.0000
    60.0010 60.0010 60.0000 59.9990 60.0000 60.0010 59.9990
    60.0000 59.9990 60.0010 60.0000 59.9990 60.0010 60.0000
    59.9990 60.0010 60.0000 59.9990 60.0010 60.0000 60.0010
    59.9990  60.0010];

cub =[60.0010 60.0010 60.0020 60.0010 60.0010 60.0030 60.0010
    60.0010 60.0030 60.0020 60.0030 60.0010 60.0020 60.0030
    60.0010 60.0020 60.0010 60.0030 60.0030 60.0010 60.0020
    60.0010 60.0020 60.0030 60.0010 60.0020 60.0030 60.0010
    60.0020];
```

Introducing this data in Matlab and assigning them together, the result obtained is:

```
The number of final pieces that are obtained is 26.

mat_sol =

  Columns 1 through 8

   25.0000   25.0030   22.0040   21.9990   41.9980   42.0060   59.9990
       60.0010

   25.0000   25.0040   22.0030   21.9990   41.9980   42.0060   60.0010
       60.0030

   24.9990   25.0020   22.0020   21.9990   42.0000   42.0070   60.0000
       60.0020

   24.9980   25.0030   22.0020   21.9990   42.0000   42.0070   60.0000
       60.0020

   25.0000   25.0040   22.0040   21.9990   41.9990   42.0070   60.0010
       60.0030

   25.0000   25.0030   22.0030   21.9990   42.0000   42.0080   59.9990
       60.0010

   24.9990   25.0040   22.0040   21.9990   41.9980   42.0060   59.9990
       60.0010

   24.9980   25.0030   22.0020   21.9980   42.0000   42.0070   60.0010
       60.0030

   24.9970   25.0020   22.0040   22.0000   41.9990   42.0070   59.9990
       60.0010

   25.0000   25.0030   22.0030   21.9990   42.0000   42.0070   60.0010
       60.0030

   24.9990   25.0040   22.0040   22.0000   41.9980   42.0060   60.0000
       60.0010

   25.0000   25.0040   22.0030   22.0000   41.9980   42.0060   60.0000
       60.0020

   24.9980   25.0030   22.0040   21.9990   42.0000   42.0080   60.0000
       60.0020

   24.9970   25.0020   22.0040   21.9990   41.9990   42.0070   59.9990
       60.0010

   25.0000   25.0030   22.0040   22.0000   42.0000   42.0070   60.0010
       60.0030

   25.0000   25.0040   22.0040   22.0000   42.0000   42.0080   60.0000
       60.0020

   24.9990   25.0040   22.0040   22.0000   42.0000   42.0070   60.0000
       60.0010
```

```
24.9970    25.0020    22.0020    21.9980    42.0000    42.0070    60.0000
    60.0010

25.0000    25.0030    22.0020    21.9990    41.9990    42.0070    59.9990
    60.0010

25.0000    25.0030    22.0040    22.0000    42.0000    42.0080    60.0010
    60.0030

24.9990    25.0030    22.0040    21.9990    41.9990    42.0070    60.0000
    60.0010

25.0000    25.0040    22.0030    21.9980    42.0000    42.0080    60.0010
    60.0020

24.9990    25.0020    22.0020    21.9990    41.9990    42.0070    60.0000
    60.0010

24.9980    25.0030    22.0030    21.9990    42.0000    42.0070    60.0010
    60.0030

24.9990    25.0030    22.0040    22.0000    41.9980    42.0060    60.0010
    60.0020

25.0000    25.0040    22.0040    22.0000    41.9990    42.0070    60.0000
    60.0010
```

This means that of the 29 possible combination, 26 spindles are achieved.

It is now studying what happens if, prior to implementing the assignment program, the optimum diameter of one element, for example the rear shaft is calculated and the pieces, according to this diameter, are produced.

Introducing the 29 rear bearings in the calculation program what is got is:

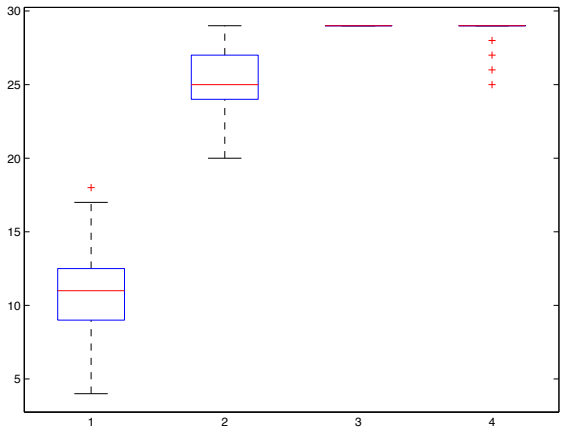


Figure 39: Obtained box plot

ANOVA Table

Source	SS	df	MS	F	Prob>F
Columns	110702.8	3	36900.9	12336.27	0
Error	5970.6	1996	3		
Total	116673.4	1999			

Figure 40: Obtained ANOVA

At first glance, in the box-plot of Figure 39, it may seem that for simulation 3 (diameter 22.003) and for the simulation 4 (diameter 22.004) the same mean value is obtained. However, in the output window can be seen that the following is obtained:

Listing 4.2: Results obtained

```
Mean value for diameter 22.001: 10.9000

Mean value for diameter 22.002: 25.5000

Mean value for diameter 22.003: 29

Mean value for diameter 22.004: 28.8300
```

Another parameter that allows to confirm that at least two mean values are different is the p-value obtained after ANOVA, shown in Figure 40, which takes a value of 0. As explained in the section of theory, a p-value smaller than 0.05 means that the results are statistically significant, meaning there is a strong evidence to suggest the null hypothesis H_0 is false. The null hypothesis states that the mean of all the groups studied are equal. With the p-value obtained in this case it can be said, therefore, that at least one of the means being studied is different from the others.

The result obtained means that the diameter to be produced, in order to get the greatest number of possible assignments, is the diameter 22.003 mm.

Once these shafts are produced, theirs real dimensions are taken. As the example is trying to be as close to reality as possible, 29 rear shafts are simulated, with a mean value of 22.003 mm and the already calculated deviation (σ = 0,00042166).

Listing 4.3: Matlab's code to carry out the simulation of diameter 22.004

```
rear_shaft=randn(29,1).*0.00042166+22.003

rear_shaft=[22.003 22.004  22.0030  22.0030  22.0020  22.0040
   22.0030  22.0030 22.0030 22.0030  22.0030 22.0020  22.0030
   22.0020  22.0020  22.0030 22.0030   22.0030 22.0030  22.0030
     22.0030  22.0030  22.0030  22.0030 22.0030 22.0030
   22.0030  22.0030  22.0020];
```

Now the allocation between all pieces is made.

```
The number of final pieces that are obtained is 26.

mat_sol =

  Columns 1 through 8

   25.0000   25.0030   22.0040   21.9990   41.9980   42.0060   59.9990
       60.0010

   25.0000   25.0040   22.0030   21.9990   41.9980   42.0060   60.0010
       60.0030

   24.9990   25.0020   22.0020   21.9990   42.0000   42.0070   60.0000
       60.0020

   24.9980   25.0030   22.0020   21.9990   42.0000   42.0070   60.0000
       60.0020

   25.0000   25.0040   22.0040   21.9990   41.9990   42.0070   60.0010
       60.0030

   25.0000   25.0030   22.0030   21.9990   42.0000   42.0080   59.9990
       60.0010

   24.9990   25.0040   22.0040   21.9990   41.9980   42.0060   59.9990
       60.0010

   24.9980   25.0030   22.0020   21.9980   42.0000   42.0070   60.0010
       60.0030

   24.9970   25.0020   22.0040   22.0000   41.9990   42.0070   59.9990
       60.0010

   25.0000   25.0030   22.0030   21.9990   42.0000   42.0070   60.0010
       60.0030

   24.9990   25.0040   22.0040   22.0000   41.9980   42.0060   60.0000
       60.0010

   25.0000   25.0040   22.0030   22.0000   41.9980   42.0060   60.0000
       60.0020

   24.9980   25.0030   22.0040   21.9990   42.0000   42.0080   60.0000
       60.0020

   24.9970   25.0020   22.0040   21.9990   41.9990   42.0070   59.9990
       60.0010

   25.0000   25.0030   22.0040   22.0000   42.0000   42.0070   60.0010
       60.0030

   25.0000   25.0040   22.0040   22.0000   42.0000   42.0080   60.0000
       60.0020

   24.9990   25.0040   22.0040   22.0000   42.0000   42.0070   60.0000
       60.0010

   24.9970   25.0020   22.0020   21.9980   42.0000   42.0070   60.0000
       60.0010
```

```
25.0000    25.0030    22.0020    21.9990    41.9990    42.0070    59.9990
    60.0010

25.0000    25.0030    22.0040    22.0000    42.0000    42.0080    60.0010
    60.0030

24.9990    25.0030    22.0040    21.9990    41.9990    42.0070    60.0000
    60.0010

25.0000    25.0040    22.0030    21.9980    42.0000    42.0080    60.0010
    60.0020

24.9990    25.0022    22.0020    21.9990    41.9990    42.0070    60.0000
    60.0010

24.9980    25.0030    22.0030    21.9990    42.0000    42.0070    60.0010
    60.0030

24.9990    25.0030    22.0040    22.0000    41.9980    42.0060    60.0010
    60.0020

25.0000    25.0040    22.0040    22.0000    41.9990    42.0070    60.0000
    60.0010
```

As can be seen, the calculation of a single optimal dimension entails no improvement in this case, as the same number of spindles built is obtained.

Now it's checked what happens if the optimum size of all parts related to the bearings (shafts and housing) is calculated and they are generated according to this dimension.

Listing 4.4: Nominal dimensions obtained

```
front_shaft=25.002 mm
rear_shaft=22.003 mm
carc_int=42.006 mm
```

After simulating, the data that are obtained are:

Listing 4.5: Matlab's code to carry out the simulation of pieces

```
front_shafts=randn(29,1).*0.000439427+25.002;

front_shafts=[25.0020 25.0020 25.0020 25.0020 25.0020 25.0020
    25.0020 25.0020 25.0020   25.0020 25.0020 25.0020  25.0020
    25.0020  25.0020  25.0030 25.0030 25.0020  25.0020  25.0020
      25.0020  25.0020  25.0020 25.0020 25.0020  25.0020
    25.0020  25.0020  25.0020];

carc_int=randn(29,1).*0.000391888+42.002;

carc_int=[42.006 42.0050 42.0060 42.0060  42.0060  42.0060
    42.0060 42.0060 42.0060 42.0070  42.0060 42.0050  42.0060
    42.0060  42.0050 42.0060 42.0060 42.0070 42.0060  42.0060
    42.0060  42.0060 42.0060 42.0060 42.0060 42.0070 42.0060
    42.0050 42.0050];
```

With these data the allocation is carried out:

```
The number of final pieces that are obtained is 29.

mat_sol =

  Columns 1 through 8

   25.0000   25.0030   22.0020   21.9990   41.9990   42.0060   60.0000
       60.0010

   25.0000   25.0030   22.0020   21.9980   41.9990   42.0060   59.9990
       60.0010

   24.9990   25.0020   22.0030   22.0000   41.9970   42.0050   60.0000
       60.0020
```

```
24.9990    25.0020    22.0030    21.9980    41.9990    42.0060    59.9990
    60.0010

24.9990    25.0020    22.0030    21.9980    41.9990    42.0060    60.0010
    60.0030

24.9990    25.0020    22.0030    21.9990    41.9990    42.0060    60.0010
    60.0020

24.9970    25.0020    22.0030    21.9990    41.9990    42.0060    60.0010
    60.0020

24.9980    25.0020    22.0030    21.9990    41.9990    42.0060    60.0010
    60.0020

24.9980    25.0020    22.0030    21.9990    41.9990    42.0060    60.0010
    60.0020

24.9980    25.0020    22.0030    21.9990    41.9990    42.0060    60.0010
    60.0030

24.9980    25.0020    22.0030    21.9990    41.9990    42.0060    60.0010
    60.0030

24.9980    25.0020    22.0030    21.9990    41.9990    42.0060    60.0010
    60.0030

24.9980    25.0020    22.0030    21.9990    41.9990    42.0060    60.0010
    60.0030

24.9980    25.0020    22.0030    21.9990    41.9990    42.0060    59.9990
    60.0010

24.9980    25.0020    22.0030    21.9990    41.9990    42.0060    59.9990
    60.0010

24.9980    25.0020    22.0030    22.0000    41.9980    42.0060    59.9990
    60.0010

24.9980    25.0020    22.0030    22.0000    41.9980    42.0060    59.9990
    60.0010

24.9980    25.0020    22.0030    22.0000    41.9980    42.0060    60.0000
    60.0010

24.9980    25.0020    22.0030    22.0000    41.9980    42.0060    60.0000
    60.0010

24.9980    25.0020    22.0030    22.0000    41.9980    42.0060    59.9990
    60.0010

24.9980    25.0020    22.0030    22.0000    42.0000    42.0060    60.0000
    60.0010

24.9980    25.0020    22.0030    22.0000    42.0000    42.0070    60.0000
    60.0010

24.9980    25.0020    22.0040    22.0000    42.0000    42.0070    60.0000
    60.0010

24.9980    25.0020    22.0040    21.9990    41.9980    42.0060    60.0010
```

```
    60.0030

24.9980   25.0020   22.0020   21.9990   41.9990   42.0050   60.0000
    60.0010

24.9980   25.0020   22.0020   21.9990   41.9990   42.0050   60.0000
    60.0020

24.9980   25.0020   22.0020   21.9990   41.9990   42.0050   60.0000
    60.0020

24.9980   25.0020   22.0030   21.9980   41.9990   42.0050   60.0000
    60.0020

24.9980   25.0020   22.0030   22.0000   42.0000   42.0070   60.0000
    60.0010
```

4.2 Analysis of the results

In the previous section has been carried out a collection of some results obtained through an example with real data from the company. This has been conducted in order to verify that the procedure developed throughout the project has been carried out correctly and that the results are the ones expected.

First it is performed the combination of parts by simply applying the allocation between them. These parts are produced in the company seeking an intermediate tolerance, but without producing them according to the bearings received and used.

By simply using the program for the combination of parts, it is already taking place a big improvement, because until now there was a lack of method in this regard. It was necessary to buy more than double of the bearings finally used. This represents a saving of nearly 50% of the money invested until now in the purchase of bearings. It is impossible to compare the results that would be obtained in the case of not using the program and using it, as in the case of not using it it's a completely random event. The workers would try to combine pieces randomly until they fit.

For this combination it's obtained a result of 26 spindles properly produced.

In the following case the combination of parts is studied when only one of these parts has been produced according to the bearings to be used. This means that prior to production, the optimum nominal dimension has

been calculated and that the parts have been produced according to this dimension.

For this combination there are no improvements over the previous case and the same spindles properly produced, 26, are obtained.

In the last case under study the three pieces that are related to the bearings are produced as a function thereof. The nominal dimensions of the parts have been estimated and they are manufactured trying to obtain such dimension.

Here the results are unbeatable because of 29 possible spindles that could occur, 29 are obtained.

Thus, after collecting and analyzing these practical results, it can be concluded and stated that the implementation of this modus operandi by the company could make a big improvement in their production process.

4.3 Benefits for the company

In the introduction it has been said that the aim of this project was to solve two problems that exist in the factory when manufacturing and assembling the spindles.

Besides solving both problems, a software that allows them to perform the mathematical resolution has been developed so that the problem can be solved from a practical and easy point of view, without the need of a mathematical knowledge. Now, this software must be implemented in the factory.

However, the results obtained after this implementation are not included in this project, since the data are available some time later.

What is going to be discussed therefore are the benefits that are expected to be obtained. Since the methods that were conducted until now were inadequate methods, there is a large room for improvement.

The main benefit for the company is that ,through an accurate coordination, a better availability of the parts required to build the spindle can be ensured. Thereby, the delivery times of Precise products will be reduced and a greater customer's satisfaction will be achieved.

The company could also improve the production's planning and the purchasing of parts, as they will know more specifically how many pieces should be produced and, moreover, of which size. This will facilitate the work of the machine's programming, thus reducing the time required for it. That means that an optimization will occur in the use of machinery.

In short, it can be said that improvements are expected to be achieved in many areas.

In the economic field, as it won't be needed to buy as many bearings as nowadays. As it has been seen, with the implementation of the program almost 100% of the bearings bought, as well as almost all of the parts manufactured, can be used. Currently more than double the bearing that they are finally used must be purchased.

In the field of quality, as more optimal spindles will be produced. Until now, what is done when manufacturing parts, was to manufacture them seeking for an intermediate tolerance. The use of this software allows to know in advanced which is the measure to be produced in order to get the greatest number of optimal spindles as a function of the bearings that have been purchased.

In addition to the improvement in the utilization of raw materials, the use of this program would allow them to save considerable time, as the only thing that the worker should spend time in is in the introduction of data. Once that is done, he should just combine the pieces according to the results shown.

Chapter 5

Conclusions

5.1 Final conclusions

The Company's chiefs were the ones who raised the existing problems as they meant important problems for them. Sometimes a lack of parts affected significantly the production chain, so a way to prevent this was sought. Also a large expenditure on purchases occurred which was higher than necessary, because more than half of the bought bearings were finally not used.

What was sought was the solution of these problems.

After the accomplishment of the project it can be stated that the objectives posed at the beginning of it have been achieved.

A process that allows the resolution of the problems that existed in the production process of the company has been successfully developed and it has been proven, that it works properly.

Moreover, a tool for the implementation of the achieved solution in the company has been programmed, so the problem can be solved in a practical and not just theoretically way.

5.2 Possible improvements

Although the initial objectives have been achieved, this does not mean that improvements can not been carried out.

It would have been very interesting to know how they performed at the

factory the application of the solution to the problem, as this application is what allows to know the failures that can arise or the possible improvements that can be carried out after the implementation.

For example both the aesthetic appearance and the speed of execution of the software could be improved, simplifying the programming in Matlab.

Some improvements or expansions could also be carried out as far as the software is concerned.

If the proper functioning of the software is demonstrated, an improvement of this software could be carried out by adding, for example, the option to calculate the number of bearings that should be bought to produce a specific number of spindles. So one improvement could be a software upgrade.

Bibliography

[1] D. S. Cheema, C. (2005) *Operations Research*. Laxmi Publications.

[4] M. Law, A. (1982) *Simulation Modeling and Analysis*. McGraw-Hill Education.

[3] S. Hillier, F. (2010) *Introduction to Operations Research*. McGraw-Hill Education.

[4] C. Montgomery, D., C. Runger, G. (2013) *Applied Statistics and Probability for Engineers*. Wiley.

[7] operationsresearch.biz <`http://www.operationsresearch.biz/assignment-problem.php` > [Consulted: 7th October 2015]

[6] Burkard, R., Dell'Amico, M., Martello, S.(2012), *Assignment Problems*. Society for Industrial and Applied Mathematics.

[7] Statistical significant <`http://www.statisticallysignificantconsulting.com/Anova.htm` > [Consulted: 7th October 2015]

[8] Ayorkor, G., Stentz, A., Bernardine Dias,M.,(2007) *The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs*.Pennsylvania: Carnegie Mellon University.

[9] pages.cs.wisc.edu <`http://pages.cs.wisc.edu/~shuchi/courses/787-F09/scribe-notes/lec5.pdf` > [Consulted: 7th October 2015]

[10] C. Tulsian, P., Pandey, V. (2002), *Quantitative Techniques: Theory and Problems*, Pearson.

[11] Seiwert, N.(2014) *Flow Optimization for concurrent Multimedia Traffic in Software Defined Networks*. Master's Thesis. Saarbruecken: Saarland University.

[12] S. Altner, D.(2008) *Advancements on problems involving maximum flows*. Thesis. Georgia Institute of Technology.

[13] Jain, C.(2010) *An Approach to Efficient Network Flow Algorithm for Solving Maximum Flow Problem*. Thesis. Thapar University.

[14] Angelo IOAN, C.,(2008). *A solving method of two economical problems using linear programming in integer numbers*. Danubius University.

Part I

Appendix

Appendix A

Matlab code

A.1 Initial menu

```
function varargout = menu_inicial_en(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
                      @menu_inicial_en_OpeningFcn, ...
                   'gui_OutputFcn',  @menu_inicial_en_OutputFcn
                      , ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin
       {:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
delete( s )


function menu_inicial_en_OpeningFcn(hObject, eventdata, handles
   , varargin)
handles.output = hObject;
```

```
guidata(hObject, handles);

function varargout = menu_inicial_en_OutputFcn(hObject,
    eventdata, handles)
varargout{1} = handles.output;


function calcular_Callback(hObject, eventdata, handles)
tabla_en


function asignar_Callback(hObject, eventdata, handles)
asignacion_todas_piezas


function ayuda_Callback(hObject, eventdata, handles)
open('help_menu.pdf')
```

A.2 Calculate nominal dimension

```
function varargout = tabla_en(varargin)

% Last Modified by GUIDE v2.5 22-May-2015 12:22:37

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @tabla_en_OpeningFcn, ...
                   'gui_OutputFcn',  @tabla_en_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin
        {:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
function tabla_en_OpeningFcn(hObject, eventdata, handles,
    varargin)

handles.output = hObject;

guidata(hObject, handles);

function varargout = tabla_en_OutputFcn(hObject, eventdata,
    handles)
varargout{1} = handles.output;

function uipanel2_SelectionChangeFcn(hObject, eventdata,
    handles)
global opc_1
global opc_2
global opc_3


if hObject==handles.eje_del
    opc_1=get(hObject,'Value');
    opc_2=0;
    opc_3=0;
elseif hObject==handles.eje_tras
    opc_2=get(hObject,'Value');
    opc_1=0;
    opc_3=0;
elseif hObject==handles.lag
    opc_3=get(hObject,'Value');
    opc_2=0;
    opc_1=0;
end

set(handles.uipanel2,'visible','off');
set(handles.text1,'visible','on');
set(handles.edit_filas,'visible','on');
set(handles.crear_tabla,'visible','on');
set(handles.cargar,'visible','on');

function edit_filas_Callback(hObject, eventdata, handles)

function edit_filas_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function crear_tabla_Callback(hObject, eventdata, handles)
```

```
fil=str2double(get(handles.edit_filas,'String'));
col=1;
size_table=cell(fil,col);
size_table(:,:)={''};
set(handles.uitable1,'Data',size_table)
set(handles.uitable1,'ColumnEditable',true(1,1))


set(handles.text1,'visible','off');
set(handles.edit_filas,'visible','off');
set(handles.crear_tabla,'visible','off');
set(handles.cargar,'visible','off');
set(handles.uitable1,'visible','on');
set(handles.calcular,'visible','on');

function cargar_Callback(hObject, eventdata, handles)
global lager

[filename pathname] = uigetfile ({'*.txt'; '*.dat'});
data = load([pathname,filename]);

num_data = length(data);
lager = data(:,1);
compl=zeros(num_data,1);
data=[lager compl];
col=2;
size_table=cell(num_data,col);
size_table(:,:)={''};
set(handles.uitable1,'Data',size_table)
set(handles.uitable1,'ColumnEditable',true(1,1))
set(handles.uitable1,'data',data)

set(handles.text1,'visible','off');
set(handles.edit_filas,'visible','off');
set(handles.crear_tabla,'visible','off');
set(handles.cargar,'visible','off');
set(handles.uitable1,'visible','on');
set(handles.calculate_cargar,'visible','on');


function calculate_cargar_Callback(hObject, eventdata, handles)

global opc_1
global opc_2
global opc_3


lager=get(handles.uitable1,'data');
```

```
%para el eje delantero 25 cm

if opc_1==1

fprintf(' \n ');
dec=0;
num_sim=500;
c=4;
mat_ana=zeros(num_sim,c);

acum_fin=0;
sol_fin=0;
eje=0;

for c=1:4
    acum=0;
    pos=0;
    sol=0;
   for k=1:num_sim
       handles.lager=lager;
       N=numel(handles.lager);
% se puede mover entre 25.001 y 25.004
mean_d3=25.001+dec
stdd_d3=0.000439427;
d3=randn(N,1).*stdd_d3+mean_d3;

d3_r=round(d3*1000)/1000;
welle=[d3_r];

m=numel(welle);
n=numel(handles.lager);
matrix=zeros(m,n);

for i=1:m
  for j=1:n
    matrix(i,j)=welle(j)-handles.lager(i);
  end
end

matrix;

mat_op=zeros(m,n);

for i=1:m
  for j=1:n
    x=round(matrix(i,j)*1000)/1000;
    if (x==0.005)
     mat_op(i,j)=0;
    end
```

```
    if (x==0.004)
      mat_op(i,j)=0;
    end
    if (x==0.003)
      mat_op(i,j)=0;
    end
    if (x>0.005)
      mat_op(i,j)=Inf;
    end
    if (x<0.003)
      mat_op(i,j)=Inf;
    end
  end
end

Matching = Hungarian(mat_op);

opt=0;
for i=1:m
  for j=1:n
    if (mat_op(i,j)==0) &&(Matching(i,j)==1)
      opt=opt+1;
    end
  end
end

kein=N-opt;

mat_ana(k,c)=sum;
if sum>acum
   acum=sum;
   pos=k;
end
  end
fprintf(' \n ');
   dec=dec+0.001;

  if acum>acum_fin
       acum_fin=acum;
       sol_fin=sol;
  end
end

[Pr, ANOVATABr, STATSr]=anova1(mat_ana);
[cr,mr,hr,nmar]=multcompare(STATSr);
boxplot(mat_ana,{'25.001','25.002','25.003','25.004'})
saveas(gcf,'boxplot.pdf')

media_fin=0;
```

```
for i=1:c
    suma_fila=0;
    cont=0;
    for j=1:k
        suma_fila(c)=mat_ana(j,i)+cont;
        cont=suma_fila(c);
    end
    suma_fin=suma_fila(c)/k
    if suma_fin>media_fin
        media_fin=suma_fin
        eje=i;
    end
end

sol_eje=0;
fprintf(' \n ');
disp('La mejor combinacion se obtiene para:');
fprintf(' \n ');
if eje==1
   disp('eje de diametro 25.001');
   sol_eje=25.001;
end

if eje==2
   disp('eje de diametro 25.002');
   sol_eje=25.002;
end

if eje==3
   disp('eje de diametro 25.003');
   sol_eje=25.003;
end

if eje==4
   disp('eje de diametro 25.004');
   sol_eje=25.004;
end
fprintf(' \n ');
sol_eje;

boxplot(handles.axes1,mat_ana)

set(handles.axes1,'visible','on');
set(handles.text2,'visible','on');
set(handles.text3,'visible','on');
set(handles.text3,'string', sol_eje);

end
```

```
%para el eje trasero de 22cm

if opc_2==1

fprintf(' \n ');
dec=0;
num_sim=500;
c=4;
mat_ana=zeros(num_sim,c);

acum_fin=0;
sol_fin=0;
eje=0;

for c=1:4
    acum=0;
    pos=0;
    sol=0;
   for k=1:num_sim
       handles.lager=lager;
       N=numel(handles.lager);
% se puede mover entre 22.001 y 22.004
mean_d3=22.001+dec
stdd_d3=0.00042166;
d3=randn(N,1).*stdd_d3+mean_d3;

d3_r=round(d3*1000)/1000;
welle=[d3_r];

m=numel(welle);
n=numel(handles.lager);
matrix=zeros(m,n);

for i=1:m
  for j=1:n
    matrix(i,j)=welle(j)-handles.lager(i);
  end
end

mat_op=zeros(m,n);

for i=1:m
  for j=1:n
    x=round(matrix(i,j)*1000)/1000;
    if (x==0.005)
     mat_op(i,j)=0;
    end
    if (x==0.004)
      mat_op(i,j)=0;
```

```
    end
    if (x==0.003)
      mat_op(i,j)=0;
    end
    if (x>0.005)
      mat_op(i,j)=Inf;
    end
    if (x<0.003)
      mat_op(i,j)=Inf;
    end
  end
end

Matching = Hungarian(mat_op);

opt=0;

for i=1:m
  for j=1:n
    if (mat_op(i,j)==0) &&(Matching(i,j)==1)
      opt=opt+1;
    end
  end
end

opt
kein=N-opt

sum=opt;

mat_ana(k,c)=sum;
if sum>acum
   acum=sum;
   pos=k;
end
end

fprintf(' \n ');
   dec=dec+0.001;
pos

end


media_fin=0;
for i=1:c
    suma_fila=0;
    cont=0;
    for j=1:k
```

```
        suma_fila(c)=mat_ana(j,i)+cont;
        cont=suma_fila(c);
    end
    suma_fin=suma_fila(c)/k
    if suma_fin>media_fin
        media_fin=suma_fin
        eje=i;
    end
end

mat_ana
[Pr, ANOVATABr, STATSr]=anova1(mat_ana);
[cr,mr,hr,nmar]=multcompare(STATSr);
boxplot(mat_ana,{'22.001','22.002','22.003','22.004'})
saveas(gcf,'boxplot_tras.pdf')

sol_eje=0;
fprintf(' \n ');
disp('La mejor combinacion se obtiene para:');
fprintf(' \n ');
if eje==1
   disp('eje de diametro 22.001');
   sol_eje=22.001;
end

if eje==2
   disp('eje de diametro 22.002');
   sol_eje=22.002;
end

if eje==3
   disp('eje de diametro 22.003');
   sol_eje=22.003;
end

if eje==4
   disp('eje de diametro 22.004');
   sol_eje=22.004;
end
fprintf(' \n ');
sol_eje;


boxplot(handles.axes1,mat_ana)

set(handles.axes1,'visible','on');
set(handles.text2,'visible','on');
set(handles.text3,'visible','on');
set(handles.text3,'string', sol_eje);
```

```
end

%para el lagerschild

if opc_3==1

fprintf(' \n ');
dec=0;
num_sim=500;
c=3;
mat_ana=zeros(num_sim,c);

acum_fin=0;
sol_fin=0;
eje=0;

for c=1:3
    acum=0;
    pos=0;
    sol=0;
   for k=1:num_sim
       handles.lager=lager;
       N=numel(handles.lager);
% se puede mover entre 42.006 y 42.008
mean_d3=42.006+dec
stdd_d3=0.000391888;
d3=randn(N,1).*stdd_d3+mean_d3;

d3_r=round(d3*1000)/1000;
schild=[d3_r];

m=numel(schild);
n=numel(handles.lager);
matrix=zeros(m,n);

for i=1:m
  for j=1:n
    matrix(i,j)=schild(j)-handles.lager(i);
  end
end

matrix;

mat_op=zeros(m,n);

for i=1:m
  for j=1:n
    x=round(matrix(i,j)*1000)/1000;
```

```
    if (x==0.006)
     mat_op(i,j)=0;
    end
    if (x==0.007)
      mat_op(i,j)=0;
    end
    if (x==0.008)
      mat_op(i,j)=0;
    end
    if (x>0.008)
      mat_op(i,j)=Inf;
    end
    if (x<0.006)
      mat_op(i,j)=Inf;
    end
  end
end

Matching = Hungarian(mat_op);


opt=0;

for i=1:m
  for j=1:n
    if (mat_op(i,j)==0) &&(Matching(i,j)==1)
      opt=opt+1;
    end
  end
end

opt
kein=N-opt

sum=opt;

mat_ana(k,c)=sum;
if sum>acum
   acum=sum;
   pos=k;
end
end

fprintf(' \n ');
   dec=dec+0.001;
pos

end
```

```
media_fin=0;
for i=1:c
    suma_fila=0;
    cont=0;
    for j=1:k
        suma_fila(c)=mat_ana(j,i)+cont;
        cont=suma_fila(c);
    end
    suma_fin=suma_fila(c)/k
    if suma_fin>media_fin
        media_fin=suma_fin
        eje=i;
    end
end

mat_ana
[Pr, ANOVATABr, STATSr]=anova1(mat_ana);
[cr,mr,hr,nmar]=multcompare(STATSr);
boxplot(mat_ana,{'42,006','42,007','42,008'})

sol_eje=0;
fprintf(' \n ');
disp('La mejor combinacion se obtiene para:');
fprintf(' \n ');
if eje==1
   disp('eje de diametro 42.006');
   sol_eje=42.006;
end

if eje==2
   disp('eje de diametro 42.007');
   sol_eje=42.007;
end

if eje==3
   disp('eje de diametro 42.008');
   sol_eje=42.008;
end

fprintf(' \n ');
sol_eje;

boxplot(handles.axes1,mat_ana)

set(handles.axes1,'visible','on');
set(handles.text2,'visible','on');
set(handles.text3,'visible','on');
set(handles.text3,'string', sol_eje);
```

```
end

function calcular_Callback(hObject, eventdata, handles)

global opc_1
global opc_2
global opc_3


c=get(handles.uitable1,'data');
lager=str2double(c(:,1));

%para guardar archivos seleccionando el nombre y la carpeta
[filename pathname] = uiputfile({'*.txt'; '*.dat'});
if filename==0
    return;
else
    fid=fopen([pathname, filename],'w');
    fprintf(fid,'%d\n%d\n',lager(:,1));
    fclose(fid);
end

%para el eje delantero 25 cm

if opc_1==1

fprintf(' \n ');
dec=0;
num_sim=30;
c=4;
mat_ana=zeros(num_sim,c);

acum_fin=0;
sol_fin=0;
eje=0;

for c=1:4
    acum=0;
    pos=0;
    sol=0;
   for k=1:num_sim
       handles.lager=lager;
       N=numel(handles.lager);
% se puede mover entre 25.001 y 25.004
mean_d3=25.001+dec
stdd_d3=0.000439427;
d3=randn(N,1).*stdd_d3+mean_d3;
```

```
d3_r=round(d3*1000)/1000;
welle=[d3_r];

m=numel(welle);
n=numel(handles.lager);
matrix=zeros(m,n);

for i=1:m
  for j=1:n
    matrix(i,j)=welle(j)-handles.lager(i);
  end
end

mat_op=zeros(m,n);

for i=1:m
  for j=1:n
    x=round(matrix(i,j)*1000)/1000;
    if (x==0.005)
     mat_op(i,j)=0;
    end
    if (x==0.004)
      mat_op(i,j)=0;
    end
    if (x==0.003)
      mat_op(i,j)=0;
    end
    if (x>0.005)
      mat_op(i,j)=Inf;
    end
    if (x<0.003)
      mat_op(i,j)=Inf;
    end
  end
end

Matching = Hungarian(mat_op);


opt=0;
for i=1:m
  for j=1:n
    if (mat_op(i,j)==0) &&(Matching(i,j)==1)
      opt=opt+1;
    end
  end
end

opt
```

```
kein=N-opt

sum=opt;

mat_ana(k,c)=sum;
if sum>acum
   acum=sum;
   pos=k;
end
  end
fprintf(' \n ');
   dec=dec+0.001;
pos

  if acum>acum_fin
       acum_fin=acum;
       sol_fin=sol;
  end
end


mat_ana
[Pr, ANOVATABr, STATSr]=anova1(mat_ana);
[cr,mr,hr,nmar]=multcompare(STATSr);
boxplot(mat_ana,{'25.001','25.002','25.003','25.004'})

media_fin=0;
for i=1:c
    suma_fila=0;
    cont=0;
    for j=1:k
        suma_fila(c)=mat_ana(j,i)+cont;
        cont=suma_fila(c);
    end
    suma_fin=suma_fila(c)/k
    if suma_fin>media_fin
        media_fin=suma_fin
        eje=i;
    end
end


sol_eje=0;
fprintf(' \n ');
disp('La mejor combinacion se obtiene para:');
fprintf(' \n ');
if eje==1
   disp('eje de diametro 25.001');
```

```
   sol_eje=25.001;
end

if eje==2
   disp('eje de diametro 25.002');
   sol_eje=25.002;
end

if eje==3
   disp('eje de diametro 25.003');
   sol_eje=25.003;
end

if eje==4
   disp('eje de diametro 25.004');
   sol_eje=25.004;
end
fprintf(' \n ');
sol_eje;

boxplot(handles.axes1,mat_ana)

set(handles.axes1,'visible','on');
set(handles.text2,'visible','on');
set(handles.text3,'visible','on');
set(handles.text3,'string', sol_eje);

end

%para el eje trasero de 22cm

if opc_2==1

fprintf(' \n ');
dec=0;
num_sim=30;
c=4;
mat_ana=zeros(num_sim,c);


acum_fin=0;
sol_fin=0;
eje=0;

for c=1:4
    acum=0;
    pos=0;
    sol=0;
   for k=1:num_sim
```

```
       handles.lager=lager;
       N=numel(handles.lager);
% se puede mover entre 22.001 y 22.004
mean_d3=22.001+dec
stdd_d3=0.00042166;
d3=randn(N,1).*stdd_d3+mean_d3;

d3_r=round(d3*1000)/1000;
welle=[d3_r];

m=numel(welle);
n=numel(handles.lager);
matrix=zeros(m,n);

for i=1:m
  for j=1:n
    matrix(i,j)=welle(j)-handles.lager(i);
  end
end

mat_op=zeros(m,n);

for i=1:m
  for j=1:n
    x=round(matrix(i,j)*1000)/1000;
    if (x==0.005)
     mat_op(i,j)=0;
    end
    if (x==0.004)
      mat_op(i,j)=0;
    end
    if (x==0.003)
      mat_op(i,j)=0;
    end
    if (x>0.005)
      mat_op(i,j)=Inf;
    end
    if (x<0.003)
      mat_op(i,j)=Inf;
    end
  end
end

Matching = Hungarian(mat_op);

opt=0;

for i=1:m
  for j=1:n
```

```
    if (mat_op(i,j)==0) &&(Matching(i,j)==1)
      opt=opt+1;
    end
  end
end

opt
kein=N-opt

sum=opt;

mat_ana(k,c)=sum;
if sum>acum
   acum=sum;
   pos=k;
end
end

fprintf(' \n ');
   dec=dec+0.001;
pos

end


media_fin=0;
for i=1:c
    suma_fila=0;
    cont=0;
    for j=1:k
        suma_fila(c)=mat_ana(j,i)+cont;
        cont=suma_fila(c);
    end
    suma_fin=suma_fila(c)/k
    if suma_fin>media_fin
        media_fin=suma_fin
        eje=i;
    end
end

mat_ana
[Pr, ANOVATABr, STATSr]=anova1(mat_ana);
[cr,mr,hr,nmar]=multcompare(STATSr);
boxplot(mat_ana,{'22.001','22.002','22.003','22.004'})

sol_eje=0;
fprintf(' \n ');
disp('La mejor combinacion se obtiene para:');
fprintf(' \n ');
```

```
if eje==1
   disp('eje de diametro 22.001');
   sol_eje=22.001;
end

if eje==2
   disp('eje de diametro 22.002');
   sol_eje=22.002;
end

if eje==3
   disp('eje de diametro 22.003');
   sol_eje=22.003;
end

if eje==4
   disp('eje de diametro 22.004');
   sol_eje=22.004;
end
fprintf(' \n ');
sol_eje;


boxplot(handles.axes1,mat_ana)

set(handles.axes1,'visible','on');
set(handles.text2,'visible','on');
set(handles.text3,'visible','on');
set(handles.text3,'string', sol_eje);

end

%para la carasa del rodamiento

if opc_3==1

fprintf(' \n ');
dec=0;
num_sim=500;
c=3;
mat_ana=zeros(num_sim,c);

acum_fin=0;
sol_fin=0;
eje=0;

for c=1:3
    acum=0;
    pos=0;
```

```
    sol=0;
   for k=1:num_sim
       handles.lager=lager;
       N=numel(handles.lager);
% se puede mover entre 42.006 y 42.008
mean_d3=42.006+dec
stdd_d3=0.000391888;
d3=randn(N,1).*stdd_d3+mean_d3;

d3_r=round(d3*1000)/1000;
schild=[d3_r];

m=numel(schild);
n=numel(handles.lager);
matrix=zeros(m,n);

for i=1:m
  for j=1:n
    matrix(i,j)=schild(j)-handles.lager(i);
  end
end

matrix;

mat_op=zeros(m,n);


for i=1:m
  for j=1:n
    x=round(matrix(i,j)*1000)/1000;
    if (x==0.006)
     mat_op(i,j)=0;
    end
    if (x==0.007)
      mat_op(i,j)=0;
    end
    if (x==0.008)
      mat_op(i,j)=0;
    end
    if (x>0.008)
      mat_op(i,j)=Inf;
    end
    if (x<0.006)
      mat_op(i,j)=Inf;
    end
  end
end

Matching = Hungarian(mat_op);
```

```
opt=0;

for i=1:m
  for j=1:n
    if (mat_op(i,j)==0) &&(Matching(i,j)==1)
      opt=opt+1;
    end
  end
end

opt
kein=N-opt

sum=opt;

mat_ana(k,c)=sum;
if sum>acum
   acum=sum;
   pos=k;
end
end

fprintf(' \n ');
   dec=dec+0.001;
pos

end


media_fin=0;
for i=1:c
    suma_fila=0;
    cont=0;
    for j=1:k
        suma_fila(c)=mat_ana(j,i)+cont;
        cont=suma_fila(c);
    end
    suma_fin=suma_fila(c)/k
    if suma_fin>media_fin
        media_fin=suma_fin
        eje=i;
    end
end

mat_ana
[Pr, ANOVATABr, STATSr]=anova1(mat_ana);
[cr,mr,hr,nmar]=multcompare(STATSr);
```

```
boxplot(mat_ana,{'42,006','42,007','42,008'})

sol_eje=0;
fprintf(' \n ');
disp('La mejor combinacion se obtiene para:');
fprintf(' \n ');
if eje==1
   disp('eje de diametro 42.006');
   sol_eje=42.006;
end

if eje==2
   disp('eje de diametro 42.007');
   sol_eje=42.007;
end


if eje==3
   disp('eje de diametro 42.008');
   sol_eje=42.008;
end

fprintf(' \n ');
sol_eje;

boxplot(handles.axes1,mat_ana)

set(handles.axes1,'visible','on');
set(handles.text2,'visible','on');
set(handles.text3,'visible','on');
set(handles.text3,'string', sol_eje);

end

function pushbutton3_Callback(hObject, eventdata, handles)
opc=questdlg('Do you want to close the program?','CLOSE','Yes',
    'No','No');
if strcmp(opc,'No')
  return;
end

if strcmp(opc,'Yes')
  close;
end

function ayuda_Callback(hObject, eventdata, handles)
open('help_calculate.pdf')
```

A.3 Assign parts

```
function varargout = asignacion_todas_piezas(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
                      @asignacion_todas_piezas_OpeningFcn, ...
                   'gui_OutputFcn',
                      @asignacion_todas_piezas_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin
       {:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

function asignacion_todas_piezas_OpeningFcn(hObject, eventdata,
    handles, varargin)
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

function varargout = asignacion_todas_piezas_OutputFcn(hObject,
    eventdata, handles)
varargout{1} = handles.output;

clear all

%creacion de la tabla

function edit1_Callback(hObject, eventdata, handles)

function edit1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end


function edit2_Callback(hObject, eventdata, handles)

function edit2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)

function edit3_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)

function edit4_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)

function edit5_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit6_Callback(hObject, eventdata, handles)

function edit6_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function pushbutton2_Callback(hObject, eventdata, handles)
```

```
%filas para el numero de rodamientos delanteros
fil=str2double(get(handles.edit1,'String'));
col=1;
size_table=cell(fil,col);
size_table(:,:)={''};
set(handles.uitable3,'Data',size_table)
set(handles.uitable3,'ColumnEditable',true(1,1))

%filas para el numero de eje trasero
fil_3=str2double(get(handles.edit2,'String'));
col=1;
size_table=cell(fil_3,col);
size_table(:,:)={''};
set(handles.uitable5,'Data',size_table)
set(handles.uitable5,'ColumnEditable',true(1,1))
set(handles.uitable4,'Data',size_table)
set(handles.uitable4,'ColumnEditable',true(1,1))

%filas para el numero de rodamientos trasero
fil_4=str2double(get(handles.edit4,'String'));
col=1;
size_table=cell(fil_4,col);
size_table(:,:)={''};
set(handles.uitable6,'Data',size_table)
set(handles.uitable6,'ColumnEditable',true(1,1))
set(handles.uitable15,'Data',size_table)
set(handles.uitable15,'ColumnEditable',true(1,1))

%filas para el numero de carcasas
fil_5=str2double(get(handles.edit5,'String'));
col_5=1;
size_table=cell(fil_5,col_5);
size_table(:,:)={''};
set(handles.uitable7,'Data',size_table)
set(handles.uitable7,'ColumnEditable',true(1,1))
set(handles.uitable17,'Data',size_table)
set(handles.uitable17,'ColumnEditable',true(1,1))

%filas para el numero de cubiertas
fil_6=str2double(get(handles.edit6,'String'));
col_6=1;
size_table=cell(fil_6,col_6);
size_table(:,:)={''};
set(handles.uitable8,'Data',size_table)
set(handles.uitable8,'ColumnEditable',true(1,1))

set(handles.uitable3,'visible','on');
set(handles.uitable4,'visible','on');
set(handles.uitable5,'visible','on');
```

```
set(handles.uitable15,'visible','on');
set(handles.uitable6,'visible','on');
set(handles.uitable7,'visible','on');
set(handles.uitable8,'visible','on');
set(handles.uitable17,'visible','on');
set(handles.text7,'visible','on');
set(handles.text8,'visible','on');
set(handles.text9,'visible','on');
set(handles.text10,'visible','on');
set(handles.text11,'visible','on');

set(handles.edit1,'visible','off');
set(handles.edit2,'visible','off');
set(handles.edit4,'visible','off');
set(handles.edit5,'visible','off');
set(handles.edit6,'visible','off');
set(handles.text1,'visible','off');
set(handles.text2,'visible','off');
set(handles.text3,'visible','off');
set(handles.text4,'visible','off');
set(handles.text5,'visible','off');

set(handles.pushbutton2,'visible','off');
set(handles.pushbutton1,'visible','on');

function pushbutton1_Callback(hObject, eventdata, handles)

global rd
global ed
global et
global rt_i
global rt_e
global carc_i
global carc_e
global cub
global t
global s

global mat_sol
global mat_sol_fin

set(handles.uitable3,'visible','off');
set(handles.uitable4,'visible','off');
set(handles.uitable5,'visible','off');
set(handles.uitable6,'visible','off');
set(handles.uitable7,'visible','off');
set(handles.uitable8,'visible','off');
set(handles.uitable15,'visible','off');
set(handles.uitable17,'visible','off');
```

```
set(handles.pushbutton1,'visible','off');
set(handles.pushbutton3,'visible','on');


rd=str2double(get(handles.uitable3,'Data'));
ed=str2double(get(handles.uitable4,'Data'));
et=str2double(get(handles.uitable5,'Data'));
rt_e=str2double(get(handles.uitable6,'Data'));
rt_i=str2double(get(handles.uitable15,'Data'));
carc_i=str2double(get(handles.uitable7,'Data'));
carc_e=str2double(get(handles.uitable17,'Data'));
cub=str2double(get(handles.uitable8,'Data'));
s=1;
t=1;
rd'
ed'
et'
rt_e'
rt_i'
carc_i'
carc_e'
cub'

set(handles.uitable9,'visible','on');
set(handles.uitable10,'visible','on');
set(handles.uitable11,'visible','on');
set(handles.uitable12,'visible','on');
set(handles.uitable13,'visible','on');
set(handles.uitable14,'visible','on');
set(handles.uitable16,'visible','on');
set(handles.uitable18,'visible','on');


valores=[s' rd' ed' et' rt_i' rt_e' carc_i' carc_e' cub' t'];

n_s=numel(s);
n_t=numel(t);
n_rd=numel(rd);
n_ed=numel(ed);
n_et=numel(et);
n_rt_i=numel(rt_i);
n_rt_e=numel(rt_e);
n_carc_i=numel(carc_i);
n_carc_e=numel(carc_e);
n_cub=numel(cub);

nodos=n_s+n_rd+n_ed+n_et+n_rt_i+n_rt_e+n_carc_i+n_carc_e+n_cub+
    n_t;
```

```
matrix=zeros(nodos,nodos);

%i=filas, j=columnas

%relacion nodo salida-rodamientos delanteros
for i=1:n_s
    for j=1:(n_rd+1)
        matrix(i,i)=0;
        matrix(i,j)=1;
    end
    for j=(n_rd+2):nodos
        matrix(i,j)=0;
    end
end


%relacion rodamientos delanteros-ejes delanteros
for i=1:n_rd
    for j=1:n_ed
        resta=ed(j)-rd(i);
        x=round(resta*1000)/1000;
        if (x>=0.003)||(x<=0.005)
            matrix(i+1,(n_rd+1)+j)=1;
        end
        if (x<0.003)||(x>0.005)
            matrix(i+1,(n_rd+1)+j)=0;
        end
    end
end

%relacion ejes delanteros
for i=1:n_ed
  matrix(i+(n_s+n_rd),(n_s+n_rd+n_ed)+i)=1;
end

%relacion ejes traseros

for i=1:n_et
    for j=1:n_rt_i
        resta=-(rt_i(j)-et(i));
        x=round(resta*1000)/1000;
        if (x>=0.003||x<=0.005)
            matrix(i+(n_s+n_rd+n_ed),j+(n_s+n_rd+n_ed+n_et))=1;
        end
        if (x<0.003||x>0.005)
            matrix(i+(n_s+n_rd+n_ed),j+(n_s+n_rd+n_ed+n_et))=0;
        end
    end
end
```

```
%relacion  rodamientos traseros
for i=1:n_rt_i
  matrix(i+(n_s+n_rd+n_ed+n_et),(n_s+n_rd+n_ed+n_et+n_rt_i)+i)
     =1;
end

%relacion rodamientos traseros-carcasa trasera
for i=1:n_rt_e
    for j=1:n_carc_i
        resta=-(rt_e(i)-carc_i(j));
        x=round(resta*1000)/1000;
        if (x>=0.006||x<=0.008)
            matrix(i+(n_s+n_rd+n_ed+n_et+n_rt_i),j+(n_s+n_rd+
                n_ed+n_et+n_rt_i+n_rt_e))=1;
        end
        if (x<0.006||x>0.008)
            matrix(i+(n_s+n_rd+n_ed+n_et+n_rt_i),j+(n_s+n_rd+
                n_ed+n_et+n_rt_i+n_rt_e))=0;
        end
    end
end

%relacion  carcasas
for i=1:n_carc_i
  matrix(i+(n_s+n_rd+n_ed+n_et+n_rt_i+n_rt_e),(n_s+n_rd+n_ed+
     n_et+n_rt_i+n_rt_e+n_carc_i)+i)=1;
end

%relacion carcasa-cubierta
for i=1:n_carc_e
    for j=1:n_cub
        resta=-(carc_e(i)-cub(j));
        x=round(resta*1000)/1000;
        if (x>=0.001||x<=0.002)
            matrix(i+(n_s+n_rd+n_ed+n_et+n_rt_i+n_rt_e+n_carc_i
                ),j+(n_s+n_rd+n_ed+n_et+n_rt_i+n_rt_e+n_carc_i+
                n_carc_e))=1;
        end
        if (x<0.001||x>0.002)
            matrix(i+(n_s+n_rd+n_ed+n_et+n_rt_i+n_rt_e+n_carc_i
                ),j+(n_s+n_rd+n_ed+n_et+n_rt_i+n_rt_e+n_carc_i+
                n_carc_e))=0;
        end
    end
end

%relacion cubierta-final
for j=1:n_cub
```

```
    suma=0;
    for i=1:n_carc_e
        suma=suma+matrix(n_s+n_rd+n_ed+n_et+n_rt_i+n_rt_e+
            n_carc_i+i,j+n_s+n_rd+n_ed+n_et+n_rt_i+n_rt_e+
            n_carc_i+n_carc_e);
    end
    if suma~=0
       matrix(j+(n_s+n_rd+n_ed+n_et+n_rt_i+n_rt_e+n_carc_i+
          n_carc_e),(n_s+n_rd+n_ed+n_et+n_rt_i+n_rt_e+n_carc_i+
          n_carc_e+n_cub+n_t))=1;
    end
end

matrix;
cm = sparse(matrix);

bg = biograph(cm,'names','LayoutType','hierarchical','
    ShowTextInNodes','Label');

[M,F,K] = graphmaxflow(cm,1,nodos);

view(bg);
solucion= biograph(F,'names','LayoutType','hierarchical','
    ShowTextInNodes','Label');
view(solucion);

mat_sol=zeros(M,9);
cont=1;
for i=1:n_rd
  from4= graphtraverse(F,i+1);
  to1= graphtraverse(F',nodos);
  h= intersect(from4,to1)
  F2 = F(h,h);
  TF = isempty(h);
  if TF==0
    mat_sol(cont,:)=h(1,:);
    if cont<M
        cont=cont+1;
    end
  end
end

mat_sol_fin=zeros(M,9);
for i=1:cont
   for j=1:9
      indice=mat_sol(i,j);
      mat_sol_fin(i,j)=valores(indice);
   end
end
```

```
set(handles.uitable11,'Data',mat_sol_fin(:,3));
set(handles.uitable16,'Data',mat_sol_fin(:,4));
set(handles.uitable12,'Data',mat_sol_fin(:,5));
set(handles.uitable13,'Data',mat_sol_fin(:,6));
set(handles.uitable10,'Data',mat_sol_fin(:,2));

set(handles.uitable9,'Data',mat_sol_fin(:,1));

set(handles.uitable18,'Data',mat_sol_fin(:,7));
set(handles.uitable14,'Data',mat_sol_fin(:,8));


[filas columnas]=size(mat_sol_fin);
set(handles.text6,'visible','on');
set(handles.edit14,'visible','on');
set(handles.edit14,'string',filas);

fprintf('El numero de piezas finales que se obtienen es %d\n',
    filas);

function edit14_Callback(hObject, eventdata, handles)

function edit14_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function pushbutton3_Callback(hObject, eventdata, handles)

global mat_sol_fin

rd=mat_sol_fin(:,1);
ed=mat_sol_fin(:,2);
et=mat_sol_fin(:,3);
rt_e=mat_sol_fin(:,4);
rt_i=mat_sol_fin(:,5);
carc_i=mat_sol_fin(:,6);
carc_e=mat_sol_fin(:,7);
cub=mat_sol_fin(:,8);

%Guardar datos en archivo EXCEL
[filename pathname] = uiputfile({'*.txt'; '*.dat';'*.xls'});
if filename==0
    return;
else
    fid=fopen([pathname, filename],'w');
```

```
    titulo = {'front bearing', 'front shaft', 'rear bearing', '
        internal rear bearing', 'external rear bearing', '
        internal bearing case', 'external bearing case', '
        housing'};
    %datos =[rd, ed, et, rt_e, rt_i, carc_i, carc_e, cub];
    %xlswrite(filename, datos);
    xlswrite(filename, titulo);
    %xlswrite(filename, datos,1, 'A2');
    %xlswrite(filename, mat_sol_fin,1, 'A2');
    %fprintf(fid,'%d\n%d\n',datos);
    fclose(fid);
end

% menu ayuda
function pushbutton4_Callback(hObject, eventdata, handles)

open('help_assign.pdf')
```

A.4 Hungarian Algorithm

```
function [Matching,Cost] = Hungarian(Perf)
%
% [MATCHING,COST] = Hungarian_New(WEIGHTS)
%
% A function for finding a minimum edge weight matching given a
    MxN Edge
% weight matrix WEIGHTS using the Hungarian Algorithm.
%
% An edge weight of Inf indicates that the pair of vertices
   given by its
% position have no adjacent edge.
%
% MATCHING return a MxN matrix with ones in the place of the
   matchings and
% zeros elsewhere.
%
% COST returns the cost of the minimum matching

% Written by: Alex Melin 30 June 2006


 % Initialize Variables
 Matching = zeros(size(Perf));

% Condense the Performance Matrix by removing any unconnected
   vertices to
```

```
% increase the speed of the algorithm

  % Find the number in each column that are connected
    num_y = sum(~isinf(Perf),1);
  % Find the number in each row that are connected
    num_x = sum(~isinf(Perf),2);

  % Find the columns(vertices) and rows(vertices) that are
     isolated
    x_con = find(num_x~=0);
    y_con = find(num_y~=0);

  % Assemble Condensed Performance Matrix
    P_size = max(length(x_con),length(y_con));
    P_cond = zeros(P_size);
    P_cond(1:length(x_con),1:length(y_con)) = Perf(x_con,y_con)
       ;
    if isempty(P_cond)
      Cost = 0;
      return
    end

    % Ensure that a perfect matching exists
      % Calculate a form of the Edge Matrix
      Edge = P_cond;
      Edge(P_cond~=Inf) = 0;
      % Find the deficiency(CNUM) in the Edge Matrix
      cnum = min_line_cover(Edge);

      % Project additional vertices and edges so that a perfect
          matching
      % exists
      Pmax = max(max(P_cond(P_cond~=Inf)));
      P_size = length(P_cond)+cnum;
      P_cond = ones(P_size)*Pmax;
      P_cond(1:length(x_con),1:length(y_con)) = Perf(x_con,
          y_con);

%************************************************
% MAIN PROGRAM: CONTROLS WHICH STEP IS EXECUTED
%************************************************
  exit_flag = 1;
  stepnum = 1;
  while exit_flag
    switch stepnum
      case 1
        [P_cond,stepnum] = step1(P_cond);
      case 2
        [r_cov,c_cov,M,stepnum] = step2(P_cond);
```

```
      case 3
        [c_cov,stepnum] = step3(M,P_size);
      case 4
        [M,r_cov,c_cov,Z_r,Z_c,stepnum] = step4(P_cond,r_cov,
            c_cov,M);
      case 5
        [M,r_cov,c_cov,stepnum] = step5(M,Z_r,Z_c,r_cov,c_cov);
      case 6
        [P_cond,stepnum] = step6(P_cond,r_cov,c_cov);
      case 7
        exit_flag = 0;
    end
  end

% Remove all the virtual satellites and targets and uncondense
    the
% Matching to the size of the original performance matrix.
Matching(x_con,y_con) = M(1:length(x_con),1:length(y_con));
Cost = sum(sum(Perf(Matching==1)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   STEP 1: Find the smallest number of zeros in each row
%           and subtract that minimum from its row
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [P_cond,stepnum] = step1(P_cond)

  P_size = length(P_cond);

  % Loop throught each row
  for ii = 1:P_size
    rmin = min(P_cond(ii,:));
    P_cond(ii,:) = P_cond(ii,:)-rmin;
  end

  stepnum = 2;

%
    **********************************************************************

%   STEP 2: Find a zero in P_cond. If there are no starred
    zeros in its
%           column or row start the zero. Repeat for each zero
%
    **********************************************************************


function [r_cov,c_cov,M,stepnum] = step2(P_cond)
```

```
% Define variables
  P_size = length(P_cond);
  r_cov = zeros(P_size,1);  % A vector that shows if a row is
     covered
  c_cov = zeros(P_size,1);  % A vector that shows if a column
     is covered
  M = zeros(P_size);        % A mask that shows if a position
     is starred or primed

  for ii = 1:P_size
    for jj = 1:P_size
      if P_cond(ii,jj) == 0 && r_cov(ii) == 0 && c_cov(jj) == 0
        M(ii,jj) = 1;
        r_cov(ii) = 1;
        c_cov(jj) = 1;
      end
    end
  end

% Re-initialize the cover vectors
  r_cov = zeros(P_size,1);  % A vector that shows if a row is
     covered
  c_cov = zeros(P_size,1);  % A vector that shows if a column
     is covered
  stepnum = 3;

%
    ************************************************************************

%   STEP 3: Cover each column with a starred zero. If all the
    columns are
%           covered then the matching is maximum
%
    ************************************************************************


function [c_cov,stepnum] = step3(M,P_size)

  c_cov = sum(M,1);
  if sum(c_cov) == P_size
    stepnum = 7;
  else
    stepnum = 4;
  end

%
    ************************************************************************
```

```
%   STEP 4: Find a noncovered zero and prime it.  If there is
    no starred
%           zero in the row containing this primed zero, Go to
    Step 5.
%           Otherwise, cover this row and uncover the column
    containing
%           the starred zero. Continue in this manner until
    there are no
%           uncovered zeros left. Save the smallest uncovered
    value and
%           Go to Step 6.
%
    ********************************************************************

function [M,r_cov,c_cov,Z_r,Z_c,stepnum] = step4(P_cond,r_cov,
    c_cov,M)

P_size = length(P_cond);

zflag = 1;
while zflag
    % Find the first uncovered zero
      row = 0; col = 0; exit_flag = 1;
      ii = 1; jj = 1;
      while exit_flag
          if P_cond(ii,jj) == 0 && r_cov(ii) == 0 && c_cov(jj)
              == 0
            row = ii;
            col = jj;
            exit_flag = 0;
          end
          jj = jj + 1;
          if jj > P_size; jj = 1; ii = ii+1; end
          if ii > P_size; exit_flag = 0; end
      end

    % If there are no uncovered zeros go to step 6
      if row == 0
        stepnum = 6;
        zflag = 0;
        Z_r = 0;
        Z_c = 0;
      else
        % Prime the uncovered zero
        M(row,col) = 2;
        % If there is a starred zero in that row
        % Cover the row and uncover the column containing the
            zero
          if sum(find(M(row,:)==1)) ~= 0
```

```
            r_cov(row) = 1;
            zcol = find(M(row,:)==1);
            c_cov(zcol) = 0;
          else
            stepnum = 5;
            zflag = 0;
            Z_r = row;
            Z_c = col;
          end
      end
end

%
    ************************************************************************

% STEP 5: Construct a series of alternating primed and starred
    zeros as
%         follows.  Let Z0 represent the uncovered primed zero
    found in Step 4.
%         Let Z1 denote the starred zero in the column of Z0 (
    if any).
%         Let Z2 denote the primed zero in the row of Z1 (there
     will always
%         be one).  Continue until the series terminates at a
    primed zero
%         that has no starred zero in its column.  Unstar each
    starred
%         zero of the series, star each primed zero of the
    series, erase
%         all primes and uncover every line in the matrix.
    Return to Step 3.
%
    ************************************************************************


function [M,r_cov,c_cov,stepnum] = step5(M,Z_r,Z_c,r_cov,c_cov)

  zflag = 1;
  ii = 1;
  while zflag
    % Find the index number of the starred zero in the column
    rindex = find(M(:,Z_c(ii))==1);
    if rindex > 0
      % Save the starred zero
      ii = ii+1;
      % Save the row of the starred zero
      Z_r(ii,1) = rindex;
      % The column of the starred zero is the same as the
          column of the
```

```
      % primed zero
      Z_c(ii,1) = Z_c(ii-1);
    else
      zflag = 0;
    end

    % Continue if there is a starred zero in the column of the
      primed zero
    if zflag == 1;
      % Find the column of the primed zero in the last starred
        zeros row
      cindex = find(M(Z_r(ii),:)==2);
      ii = ii+1;
      Z_r(ii,1) = Z_r(ii-1);
      Z_c(ii,1) = cindex;
    end
  end

  % UNSTAR all the starred zeros in the path and STAR all
    primed zeros
  for ii = 1:length(Z_r)
    if M(Z_r(ii),Z_c(ii)) == 1
      M(Z_r(ii),Z_c(ii)) = 0;
    else
      M(Z_r(ii),Z_c(ii)) = 1;
    end
  end

  % Clear the covers
  r_cov = r_cov.*0;
  c_cov = c_cov.*0;

  % Remove all the primes
  M(M==2) = 0;

stepnum = 3;

%
    ************************************************************************

% STEP 6: Add the minimum uncovered value to every element of
    each covered
%         row, and subtract it from every element of each
    uncovered column.
%         Return to Step 4 without altering any stars, primes,
    or covered lines.
%
    ************************************************************************
```

```
function [P_cond,stepnum] = step6(P_cond,r_cov,c_cov)
a = find(r_cov == 0);
b = find(c_cov == 0);
minval = min(min(P_cond(a,b)));

P_cond(find(r_cov == 1),:) = P_cond(find(r_cov == 1),:) +
    minval;
P_cond(:,find(c_cov == 0)) = P_cond(:,find(c_cov == 0)) -
    minval;

stepnum = 4;

function cnum = min_line_cover(Edge)

  % Step 2
    [r_cov,c_cov,M,stepnum] = step2(Edge);
  % Step 3
    [c_cov,stepnum] = step3(M,length(Edge));
  % Step 4
    [M,r_cov,c_cov,Z_r,Z_c,stepnum] = step4(Edge,r_cov,c_cov,M)
        ;
  % Calculate the deficiency
    cnum = length(Edge)-sum(r_cov)-sum(c_cov);
```